A "TUTOR" MINICOMPUTER SYSTEM

SofTech, Incorporated

7 February 1975

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER AD/A005 782
4. TITLE (and Subtitle) A TUTOR Minicomputer System		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER 1016
7. AUTHOR(s) John W. Brackett		8. CONTRACT OR GRANT NUMBER(s) MDA903-74-C-0288
9. PERFORMING ORGANIZATION NAME AND ADDRESS SofTech, Inc. 460 Totten Pond Road Waltham, MA 02154		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Human Resources Research Office Defense Advanced Research Projects Agncy 1400 Wilson Blvd, Arlington, VA 22209		12. REPORT DATE February 7, 1975 13. NUMBER OF PAGES 117
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) DCASR - Boston 666 Summer Street Boston, MA		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

PRICES SUBJECT TO CHANGE

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer-Aided Instruction, PLATO System, TUTOR Language

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The study examined the feasibility of a minicomputer-based system for presenting computer-aided instructional material developed in the TUTOR language currently in use on the PLATO System at the University of Illinois. The study developed a system design based upon the Digital Equipment Corporation PDP-11 computer and evaluated the costs of transferring TUTOR programs from the PLATO System to a PDP-11 based system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SofTech The Software Technology Company
460 Totten Pond Road, Waltham, Massachusetts 02154 617-890-6900

 Final Report

A TUTOR Minicomputer System

Item Number 0002AD

Contract MDA903-74-C-0288

SofTech Contract Number 1016

February 7, 1975

Sponsored By:

Advanced Research Projects Agency

ARPA Order Number 2517

Submitted By:

John Brackett
Principal Investigator
SofTech, Inc.
Waltham, Massachusetts 02154

ACKNOWLEDGEMENTS

This study was directed by John W. Brackett; the members of the project staff were Fred Moses, Lawrence Shafer, and Jorge Rodriguez.

Consulting assistance was provided by Professor Roy Kaplow of the Massachusetts Institute of Technology and by members of the PLATO Systems Group, Computer-Based Education Research Laboratory, University of Illinois. The assistance of Paul Tenczar, Bruce Sherwood, and Rick Bloome was essential to our understanding the facilities provided by the PLATO System and its performance properties. Discussion with Paul Tenczar, Professor Donald Bitzer of the University of Illinois and Ed Gardner of the Air Force Human Resources Laboratory stimulated our work and helped us focus on the critical issues of the study.

In spite of all the assistance we received, SofTech is solely responsible for this report.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF EXHIBITS

Section 1

STUDY OBJECTIVES AND CONCLUSIONS

1.1 Study Overview

With the increasing investment by the Federal Government in computer-aided instruction, considerable attention has focused on approaches to increasing significantly the number of sites at which such instructional material may be used. Many alternative approaches to providing wide-spread access to existing material have been proposed, varying from large central sites with communications networks capable of supporting hundreds or thousands of student terminals to mini-computer-based systems supporting a few student terminals. Distribution to remote, overseas or classified sites requires that alternatives to the large central site approach be considered.

The largest existing inventory of computer-aided instructional material has been developed in the TUTOR language (1) on the PLATO System (2) at the University of Illinois. Lessons in TUTOR have been written by authors from the Air Force, Army and Navy to explore the potential of the PLATO system, and of computer-aided instruction in general. However, classified material can not be developed on the PLATO System and access from overseas or remote sites is not presently feasible. Therefore, SofTech has examined the technical feasibility and cost of developing a minicomputer-based system on which new TUTOR programs could be generated and on which existing TUTOR programs could be presented or edited. If such a minicomputer-based TUTOR system were both feasible and cost-effective, use of materials developed under federal support on the PLATO System might grow substantially.

1.2 Study Tasks

The overall goal of this study was to examine the feasibility of a minicomputer-based TUTOR system based upon the Digital Equipment Corporation PDP-11 line of computers. The PDP-11 line was selected, with the agreement of ARPA, because it is a representative of a new generation of small computers which have an architecture characteristic

of larger systems and which can efficiently support software implemented in a higher-level language. The PDP-11 is in widespread use in the government and an ARPA network interface already exists.

The study evaluated how to minimize the costs of transferring TUTOR programs from the PLATO System to the PDP-11 based system. In addition, key PDP-11 hardware selection issues were examined in order to recommend how the hardware configuration should depend upon the number of active terminals to be supported.

The following tasks comprised the study:

1. Evaluate the author subset (3) of the TUTOR language to elucidate the technical impediments to implementing that TUTOR subset in a non-PLATO environment. In particular the study was to examine:
   a. Dependencies on characteristics of Control Data CYBER series computers which can affect the transferability between machines of TUTOR programs. This computer system, which is the base of the PLATO system, is the only one on which TUTOR programs currently execute.
   b. Dependencies on the facilities provided by the PLATO operating system which must be reproduced in order for a TUTOR program to perform the same actions on another computer system.
2. Develop the systems architecture for a PDP-11 based system for authoring and presenting lessons written in TUTOR.
3. Recommend PDP-11 hardware configuration for the following systems configurations:
   a. <10 simultaneously active terminals
   b. 10-30 active terminals
   c. >30 active terminals.
4. Present a plan for the detailed design, implementation and validation of the PDP-11 based system. Critical aspects of the plan were recognized to be:
   a. The approach to be used to implement the TUTOR language processor and the TUTOR run-time interpreter so that the software will be machine-independent to the maximum feasible extent.

b. Evaluate the extent to which Digital Equipment Corporation supplied software for the PDP-11 could be used to minimize the development cost and to reduce follow-on maintenance costs.

c. Develop a systems validation plan to insure that:

- TUTOR programs which execute on the PLATO system execute in an equivalent manner on the new system.
- The functional requirements of the system are met.
- The performance requirements of the system are met.

1.3 Study Assumptions

Some assumptions are basic to the approach agreed upon with ARPA at the beginning of the study. An important, but perhaps the most questionable assumption, is that a significant amount of valuable instructional material, especially material relevant to military training, will exist in TUTOR by 1977, the earliest date at which a minicomputer-based TUTOR system could be operational. If this assumption were not valid, many of the reasons would disappear for developing a minicomputer-based system around the facilities needed to support the execution of TUTOR programs. Some knowledgeable workers in the field of computer-based instructional systems question this assumption and apparently believe that only a very small fraction of the instructional material (4) on the PLATO system is sufficiently polished that conversion to another system without significant rewriting would be warranted. If few lessons will be ready for export from the PLATO system in the near future, alternative languages for developing material for the minicomputer-based system should be considered. SofTech did not evaluate the existing lesson material on the PLATO system sufficiently to be able to assess the validity of this assumption.

The plasma-panel display terminal currently used on the PLATO system is assumed to be basic to the use of TUTOR on any machine. The TUTOR language is closely integrated with the capabilities of the terminal, especially the selective erasure facility.

The validity of a minicomputer-based approach to developing and presenting instructional material requires that distributed systems have some advantages over large systems located at central sites. SofTech believes advantages exist for classified sites, remote locations with a small number of students, or locations having many students simultaneously using only a small number of lessons.

In our opinion, communications costs will not decline significantly enough within five years to make the distributed site approach economically unjustifiable. Professor Donald Bitzer of the University of Illinois, who has lead the development of the PLATO system, projects communication costs, as well as central site hardware costs, falling to such an extent that the main justification for a minicomputer-based system would have to come from classified sites, overseas sites, or sites sufficiently remote that data communication to a central site were either not feasible or were not cost effective.

SofTech has not assumed that a distributed system using minicomputers will significantly reduce the computer system costs per terminal hour currently being measured on large systems such as PLATO; in some cases the terminal hour cost may increase. The justification for the distributed system approach must come from either reduction in communications costs or from the fact that use of a large central facility such as PLATO is not viable due to one of several factors, such as the use of classified material.

1.4 Summary of Study Conclusions

The principal conclusion of the study is that a technicaliy feasible approach exists for handling 24 to 32 active terminals on a dual processor PDP-11 system with a hardware cost, excluding terminals, of about $240,000. This hardware configuration is considerably more costly than originally estimated due to the difficulty of separating the TUTOR language from the facilities provided by the PLATO system.

A small number of TUTOR/PLATO features, which will be discussed in more detail in Section 2, heavily impact the design of a system intended to provide the user with equivalent capabilities; these features are:

1. Interaction with animated displays.
2. Echoing each key typed to the display screen, especially when lesson material is required for the echoing.
3. The response time characteristics expected by current users.
4. Sharing of data among several students simultaneously working on the same lesson.
5. The use of large vocabularies for analyzing student responses.

The first three features are all different aspects of the highly interactive environment provided by the PLATO System, an environment which was assumed in writing many existing TUTOR lessons.

The decision was made at the beginning of the study, with ARPA agreement, that the system design should not be based upon eliminating any features in the author subset of TUTOR; however, the system design in Section 6 includes restrictions on:

1. The number of simultaneous users requiring lesson material for key echoing.
2. The number of TUTOR micro tables, which are used to specify graphics to be output during key echoing.
3. The number of users requiring real-time interaction with animated output.

Although some manual conversion of existing TUTOR code will be required, few programs are likely to be affected. Manual conversion of programs to be developed in the future can be nearly eliminated if a few programming standards are followed and minor enhancements are included in TUTOR to allow more machine-independent programs to be written.

A significant software development effort, which is estimated at 12-16 man years, will be required to develop the minicomputer-based system. This estimate assumes that the Digital Equipment RSX-11D operating system will provide an adequate base for the system.

Section 2

TUTOR/PLATO ANALYSIS

An important component of the design study was the evaluation of the problems to be encountered in implementing a TUTOR language processor and interpreter to operate on a computer system other than the Control Data CYBER System for which TUTOR is currently available. Section 2.1 describes the inherent machine dependencies of the TUTOR language.

There are important constraints on the operating system environment if TUTOR programs are to execute on a different computer system. The design approach taken in this study is intended to insure that the PDP-11 based system provides interactive capabilities similar to those provided by the PLATO System. Section 2.2 summarizes key statistics on the PLATO System such as interaction rates and size of programs and data required per interaction. This data has heavily influenced the software architecture for the minicomputer-based system and is presented in detail in Appendix A.

The implementation of a new TUTOR language processor and interpreter, which is to be equivalent to the version now operating at the University of Illinois, depends upon the adequate documentation on the exact meaning of the language, i.e., the language semantics. In Section 2.3 the status of this documentation is discussed and an approach is recommended for fully utilizing the expertise available within the PLATO Systems Group at the University of Illinois.

The subsequent sections of this report assume some knowledge of the PLATO System and the TUTOR language; references providing the background information required are (1, 3, 5, 6).

2.1 Machine Dependencies of TUTOR

The TUTOR language was not designed with the goal of developing machine independent software and, therefore, the characteristics of the CDC CYBER computer systems are implicit in some of the commands.

The most important machine dependencies fall into the following categories:

1. Storage allocation for character data
2. Bit manipulation operations
3. Storage of more than one data value per machine word
4. Allowable range of integers
5. Floating point precision and range
6. Character set and collating sequence assumptions

In addition, many features of the language depend upon the use of the PLATO plasma panel display terminal as the student terminal; these assumptions are not discussed here since the study assumes that all use of TUTOR-generated material will be via this type of terminal due to the many dependencies of the language on the terminal characteristics.

A basic assumption of the TUTOR design is that one machine word can store a reasonable representation of any data type. Although this is true on the CDC CYBER series due to the 60 bit word length, the assumption creates most of the machine dependencies of the language. One 60 bit word is assumed to hold either:

1. An integer value
2. A floating point value
3. Ten 60 bit characters

and the use intended by the programmer for the word is not explicitly declared in the program. The problems created by this basic assumption are very similar to those encountered when a FORTRAN program is moved from the CDC CYBER to another computer.

A key machine dependency is storage allocation for character data. Ten six-bit characters can be stored per CYBER machine word. An upper case, superscript, or subscript character requires two six-bit characters. The type of machine dependency in TUTOR can be illustrated using the MOVE command for moving character data, which has the following format:

MOVE string 1, start 1, string 2, start 2, # characters to move

where:

string 1	specifies the string from which characters are to be taken
start 1	is the character position in string 1 indicating the first character to be moved
string 2	specifies the destination string
start 2	is the character position in string 2 to which the characters will be moved

characters to move is the number of characters to be moved

Note that the MOVE command is not directly machine dependent. However, both strings are stored in variables which can hold integer, floating point or character data which is stored 10 characters per word. Therefore, the number of words affected by the MOVE is a function of the machine word length and the number of bits per character. In the TUTOR program no type declarations for character data are available and no declarations are required to indicate the number of machine words required by a character string. Therefore, a MOVE command for ten characters can have as its target a single variable on the CYBER, while on a shorter word-length machine an array would be required. The problem becomes more serious since all TUTOR variables are considered to be stored in consecutive core locations. Therefore, the MOVE command can effect variables that are not mentioned in the command, and the extent to which these other variables are affected is machine dependent.

Bit-wise operations are allowed on a full machine word or a portion of a machine word. There are bit manipulation operators in TUTOR for performing the following operations:

Shifts
Circular Shifts
Logical AND
Logical OR
Logical XOR
Counting number of bits set in a word

A complement can only be obtained by negation, which implies that ones-complement arithmetic is used by the machine. However, there are very few situations in which an author is likely to need the bit manipulation

operators since TUTOR now allows partial word values to be stored and retrieved without using the bit manipulation operators. For example, 60 logical values can be stored with distinct names in one machine word and each value can be read and set via TUTOR commands. The only situation in which an author might use the bit manipulation operators is to test or set several logical variables in one operation. Obviously, such uses of the bit manipulation operators are dependent upon the machine word length.

It is unlikely that many lessons need to make use of the bit manipulation operators and many of those that do, use them to read and set partial word values, since, until recently, these facilities were not provided in TUTOR. Prior to moving lesson material to a new system, an author would have to inspect all uses of the bit manipulation operators for uses that are machine dependent, and replace such uses with alternative TUTOR constructs. Existing programs on the PLATO System for automatically updating lesson material when the TUTOR language is changed, should make it easy to locate all uses of the bit manipulation operators in a collection of lesson material.

Arrays of integer values each requiring less than one CYBER machine word can be declared using the TUTOR DEFINE command. Such variables are defined as an array which is overlayed upon the machine words which comprise the data area of a TUTOR program. The length of the array depends upon the number of bits to be used to store each value and the number of values to be stored. Since the length is not explicit in the TUTOR program and the author is solely responsible for the layout of other variables in the data area, author errors may create storage allocation conflicts. This storage allocation problem is potentially serious if a machine with a word length significantly shorter than 60 bits is to be used to execute existing TUTOR programs.

The use of the CYBER implies that very large and very small integers and floating point values may be stored, as well as very precise floating point values. It is unclear the extent to which TUTOR programs utilize these capabilities, and, therefore, it is difficult to predict the problems which might occur if a smaller range of values or less

precise floating point values were implemented. Integer values on the CYBER are represented by 48 bits. Floating point values have 48 bits of precision and must be within the range 2^{-1024} to 2^{1023}.

Character set conventions affect the language in some ways that will cause problems, particularly the use of two characters to represent a capital letter. Many programs depend upon the two character coding for capital letters, both in terms of storage allocation and program logic. In addition, the codes from input devices are mapped into TUTOR character codes and many programs are dependent on this encoding.

The most important machine dependency appears to be storage allocation for character data, since this type of data is used in almost every TUTOR program. Moving existing programs from the PLATO environment will require some manual changes to the programs if certain TUTOR features are used, especially bit manipulation. In addition, the storage allocation of programs will have to be examined. Although some conversion aids may be helpful, the total effort, even if done manually, should not be large. Section 4 describes the approaches selected for handling the machine dependencies in the design of the PDP-11 based system.

2.2 System Dependencies of TUTOR

TUTOR was developed as a language for exploiting the facilities of the PLATO System and the plasma display terminal. The dependencies upon the plasma display terminal and its accessories such as the touch panel and audio output, are such that the characteristics of that terminal are essential to the effective use of TUTOR. The dependencies on the capabilities provided by the PLATO operating system are not as obvious, but can significantly affect the success achieved in converting existing TUTOR lesson material from the PLATO environment to another computer system.

The features of the PLATO environment upon which existing TUTOR programs appear to most heavily depend, are:

1. The ability to echo each input key to the terminal with processing prior to output.

The input of a superscript key followed by a letter can, therefore, be made to appear on the screen as a superscript. Most key echoing is done by the PLATO System routines, but author provided material can be invoked. Key echoing is on a character-by-character basis with only a fraction of a second delay before the character input is displayed on the screen, thereby providing the illusion required for touch typing of a direct connection between the keyboard and the screen.

2. The ability to interpret a response on a character-by-character basis without requiring an "end-of-input" character.

With the latter facility, authors can write highly interactive programs where every key stroke controls the lesson. For example, every time a key is pushed in the simulation of a chemical reaction, it may indicate that more liquid is to be used and the lesson will show the additional liquid on a diagram within a fraction of a second after the key is pushed. Character-by-character interpretation allows any number of function keys to be provided to the student with each key specifying a different program action.

Two less frequently used facilities, but ones which have a significant effect on the design of the minicomputer-based system, are:

3. The ability to produce animated drawings which appear to change continuously.

4. The ability to interact with animated drawings in a manner so that a lesson can determine what had been output to the screen when the student made his response.

The latter capability is used, for example, in the chemistry lesson where the student performs a simulated titration. When the student observes the titration endpoint, he must immediately stop requesting chemical to be supplied from his buret. Therefore, the success of the lesson depends upon being able to correlate what the student saw on the screen with the input he provided. These two capabilities are required to create many

simulations. Although such simulations are only a small percentage of all TUTOR lessons, they are very effective and provide one of the best justifications for a computer-based instructional system.

The list above indicates that the problem of developing a TUTOR processor for a given machine goes beyond issues that are usually considered part of a language definition, if authors familar with TUTOR are to be satisfied that the teaching material has the same "feel" to the student in the new computer system environment as it did in the PLATO environment. If the four capabilities listed above are to be provided in a minicomputer-based system, the system must support the development of extremely interactive lessons. The degree of interaction required is higher than is available in nearly any commercially available time-sharing system and SofTech knows of no system other than PLATO that supports hundreds of terminals in such an interactive manner. However, the success of the PLATO System depends upon the fact that only a very few terminals are simultaneously using the four capabilities listed above. If a majority of terminals were to run a highly interactive simulation-type lesson simultaneously, the number of terminals which could receive adequate response on the PLATO System would be much smaller than usual. The fact that most students are "thinking" or making minimal requests for service is basic to the ability of PLATO to give each student the illusion of a very interactive computer awaiting his response. However, this assumption becomes less applicable in a 20-30 terminal minicomputer-based system where only a few students interacting very heavily with the system will require an unacceptable proportion of the available computer resources.

Approaches to providing these four capabilities, at least to a smaller number of students, have significantly affected the design of the PDP-11 based system. Other key features with an important effect on the overall system design are:

1. The sharing of data (TUTOR COMMON) among multiple active students.
2. The use of large TUTOR vocabularies and a large number of TUTOR concepts for response

analysis, thereby allowing student input about a well-bounded subject area to use a free-form English sentence.

The PLATO System supports very interactive computing by exploiting a very high speed electronic swapping memory (CDC ECS memory) which allows large amounts of information to be transferred in and out of central memory where programs are executed, at speeds one or two orders of magnitude faster than the transfer rater currently achievable by drum or disk memories. ECS memory costs approximately one dollar per word and is currently available only for Control Data Corporation computers.

The extensive monitoring facilities in the PLATO System can provide a detailed characterization of how the system is used by students and authors and the size of programs and data needed per interaction to support that use. The PLATO data gathered during the study is summarized in Appendix A, but the following statistics have significantly influenced the overall design of the PDP-11 based system:

1. The average PLATO student user inputs one keystroke every four seconds.
2. Every other key press requires lessson material to process; i.e., each terminal requires use of lesson material every 8 seconds.

The latter statistic reflects the heavy use of function keys to control the lesson and indicate a marked difference between input handling on the PLATO System and on the typical time sharing system where atleast 5-10 characters will be typed before completing a command.

3. The average TUTOR lesson consists of 50-70 units (3000-4000 60 bit words).
4. The average interaction executes nine TUTOR units, of which three or four are different. Since the average TUTOR unit is about 55 60 bit words in length, about 200 words of lesson material plus associated data is required per interaction.
5. Each time a lesson material is executed, the following data is required in central memory:
   a. TUTOR student variables (150 60 bit words), which are the basic data of the lesson.

b. PLATO status data (250 60 bit words) which records the status of the lesson, student and terminal.

and the following data will be used by some lessons:

c. TUTOR COMMON data shared among active users of the lesson (up to 1500 words). Less than 25 percent of existing lessons use COMMON data.

d. Extra storage variables (up to 1500 words) which allow the basic data of the lesson to be enlarged. The sum of items c. and d. cannot exceed 1500 words. Only a few existing lessons use extra storage variables.

e. Vocabulary for response analysis (no fixed upper limit but seldom exceeds 2000-3000 words).

f. TUTOR Micro table (up to 256 words) for author-specified key echoing. A majority of lessons do not use micro tables.

6. Approximately 16,000 CYBER machine instructions are used to process an average interaction, i.e., the typical user will use about 2000 CYBER instructions per second while his terminal is active. Authors developing lesson material, or lessons requiring very heavy interaction between the student and the terminal, can use significantly more CPU time.

In summary, every 8 seconds for every terminal, the PLATO System will need to access at least 400 words of data, and possibly as many as over 4000 words of data, and approximately 200 words of program material. The program material will be much larger unless only the units actually referenced are accessed. In addition, 400-1900 words of data will need to be written out to secondary storage when processing is terminated; vocabulary data and micro tables are read-only data. The CPU requirements per average interaction are quite low. The most difficult problem involved in the design of the PDP-11 based system was to achieve the highly interactive environment which these statistics imply.

2.3 Documentation of TUTOR/PLATO

The documentation currently existing at the University of Illinois is not an adequate base for implementing a TUTOR processor for another

computer system. In particular, no language reference manual exists that defines the complete semantic effect of executing each command. Such a document will have to be developed prior to the start of implementation since a TUTOR command can affect state variables which, in turn can affect the actions resulting from subsequent commands. Much of this information can be obtained only by an in-depth examination of the assembly language listings for the TUTOR interpreter. Existing printed documentation is user-oriented and does not describe all the details associated with each command. Additional documentation on each command is available on the PLATO System as a TUTOR lesson which can be output for off-line reference.

During this study SofTech has received full cooperation from University of Illinois personnel, especially Paul Tenczar, Richard Blomme, and Bruce Sherwood. However, the PLATO project staff does not have the available time required to produce the language reference manual, but are willing to participate in such an effort by answering questions on the language semantics and reviewing draft versions of the manual. The lack of existing documentation appears the most significant technical barrier to a new implementation of TUTOR, but this problem can be overcome if the PLATO staff can provide the time required to consult on the development of the language reference manual. SofTech's best estimate is that about 6 to 9 man-months of non-Illinois labor will be required to produce the required documentation. If the key members of the PLATO project staff who have been involved with the development of TUTOR since its inception were to be unavailable, due to their other commitments or no longer being associated with the University of Illinois, SofTech doubts that reimplementation of TUTOR would be feasible.

Section 3

TESTING AND VALIDATION APPROACHES

3.1 Overview

The validation of the minicomputer-based TUTOR system must be a key element of the design approach since the system is intended to execute existing TUTOR programs in a manner "equivalent" to executing the programs on the PLATO system. A demonstration that programs behave in an "equivalent" manner depends upon establishing a suitable definition of "equivalent" and then measuring whether the behavior meets the definition.

Determining how "equivalent" should be defined is not a simple task; for example, are two response judging algorithms equivalent if only one recognizes a given word as a likely misspelling of the correct answer, although both recognize the correct spelling? Such differences are likely to occur in two different implementations and "equivalent" must be defined in a way that does not require instruction-level simulation of the CDC CYBER computer system in order for two TUTOR implementations to be considered equivalent. In addition, PLATO users may judge a new system as non-equivalent based upon performance factors, such as response time or the ability to produce animated drawings with continuous movement, even though exactly the same output is being transmitted to the terminal in both systems.

The SofTech approach to insuring that lessons execute in a equivalent manner on both systems can be divided into four stages:

1. Testing the effects of executing individual commands or a group of commands with a wide variety of inputs.
2. Testing using lessons especially prepared to exercise a wide variety of commands and to demonstrate that they interact correctly.
3. Final validation using existing TUTOR lessons and experts familiar with the lessons.
4. Performance testing.

SofTech is confident that the testing and validation approach can be refined during the detailed design stage to produce a plan that is both cost

effective and which will assure ARPA that the goal of program transferability between the two systems can be achieved. In subsequent sections we discuss in more detail some of the techniques which can be used to carry out each of the stages of testing.

During the development of the TUTOR language reference manual described in Section 2.3, a number of areas in the TUTOR language are likely to be identified where the definition of "equivalent" is not obvious. The form of equivalency for which tests are to be made will be documented as part of the manual. For example, two versions of the same judging commands might well be considered equivalent even if they differed in the words they regarded as misspellings of a valid answer.

Stage One is intended to insure to the maximum extent possible that each TUTOR command is correctly processed. However, the interaction of commands through a large number of state variables means that testing of each command in isolation is not likely to detect some errors. The lessons generated in Stage Two are intended to test the flow of control through one or more lessons while testing that commands interact correctly. The lessons used in this stage are not intended to have any educational value but will be designed to exercise the system heavily in areas that Stage One testing did not adequately cover.

The lessons used in Stage Three are intended to validate that the two systems are functionally equivalent from the viewpoint of knowledgeable PLATO users who use the lessons on both systems. When differences are observed, they will be documented and it will be decided whether the differences violate the definitions of equivalency. In some cases this type of experience may point out problems with the way in which equivalency is defined; if so, changes to the new system may be necessary to produce a closer correspondence between the effects of using the same lesson on the two systems.

The performance testing in Stage Four is designed to demonstrate that the system successfully meets its design goals and that it degrades, rather than fails, under overload. Measurements of response time for different types of inputs under various system loads will be made. In addition, the system will be forced into states which reflect unusual

requests for system resources and which would not normally be encountered. The tests may demonstrate design or implementation problems that might otherwise show up only under unusual circumstances during use.

3.2 PLATO Facilities to Support the Testing Program

The methods used to perform the first two stages of testing will depend upon the techniques available for comparing the results of executing equivalent TUTOR programs with identical input on both systems. The preferable way of obtaining a controlled input mechanism is by simulating terminal input on both systems from a disk file. Since all tests would be easily repeatable, retesting can be made more precise. In order to utilize this testing approach, modifications to the PLATO system to accept terminal-type input from a file and to write a repackaged form of output to a file would be required. Unfortunately, the cost of these modifications to the PLATO System may be prohibitive due to the manner in which terminal input and output is handled. In addition, the PLATO System group is unconvinced of the utility of a testing approach involving simulated terminal input. Therefore, the testing program described in this report depends largely upon using existing PLATO and TUTOR facilities for writing self-testing programs and for tracing execution and answer analysis during the first two stages of testing. To facilitate development testing, SofTech believes that any new system should be implemented with this type of file input although this facility may not be available on the PLATO System.

Self testing programs may be written in TUTOR using the COPY and PRESS commands to simulate input. In this way a character string stored in the TUTOR program or input from the disk can be treated as if it were input by the student. Although this mechanism requires specially written TUTOR lessons and makes it difficult to run the same lesson with different sets of inputs, it does enable test programs with carefully controlled input to be developed during the early stages of system development. This approach will be used to compare the response analysis portions of the systems with a wide variety of vocabularies and concepts as well as input responses.

PLATO also provides facilities for logging student response data, which allows tracing the flow of execution and summarizing the results of response analysis. Summary information on the number of responses judged correct or wrong can be collected without any lesson modifications. Although these facilities provide only a broad verification of actual sequencing during execution, these traces can be used to validate the sequence of answers and the corresponding system actions. Student variables and text messages can be put into the data logging file by the OUTPUT command. Therefore, by adding some OUTPUT commands to existing lessons, or to the lessons to be created during the second stage of testing, differences between the two systems in the flow of execution, response judging and how the student variables are changed may be detected. Although many of these differences might be detected by users familiar with the lessons being tested, the use of these facilities may well focus attention on problems that appear minor in the context of one lesson but which would appear in a much more noticeable form under other circumstances.

Since the PLATO System does not provide the software mechanisms for providing simulated terminal input, the only alternatives are the development of lessons using easily repeatable protocols emphasizing function key input or the use of a hardware device to provide repeatable input sequences. The simplest hardware approach is an audio tape recorder which can be used to provide all lesson input in a reproducible manner. The recording is made by a user executing the lesson and can be replayed during repetitive testing. We expect to make heavy use of such an input device during the initial stages of testing.

3.3 Comparing PLATO and the New System

A variety of methods of recording the output from the systems and comparing them in order to detect small differences have been considered, but a computerized method of doing the comparison is likely to create more problems than it solves. Recording the output to the terminal and using a computer to compare the outputs will detect a large variety of meaningless differences. For example, on each system, different sets of graphic commands are likely to be used to draw a circle, but a user will be unable to note any differences between the circles. The many meaningless differences

detected are likely to numb the testers to any significant differences that may be detected. Therefore, the most feasible comparison approach for highlighting significant difference appears to be one based upon side-by-side visual comparisons of the output from each system.

Audio tape records can be used to provide equivalent inputs to both systems and to record the resulting terminal output. Comparison of the two systems would be performed by visually observing the output on two side-by-side terminals, each driven by an audio tape recorded during the computer session. The tapes would be run step-by-step so that supposedly equivalent displays could be compared. In addition, the tapes could be run continuously to compare performance in equal time periods. The use of audio tapes is expected to have an important role in integration testing of subsystems involving terminal handling as well as during system testing since the tapes can provide controlled, repeatable input scenarios.

The final validation of the equivalence of the two systems can only be assured via a comparison of the full interactive behavior of the systems. The PLATO Systems Group recommends the use of authors and other persons familiar with selected lessons. They propose that two PLATO systems programmers run on the new system a standard lesson set consisting of approximately a dozen TUTOR lessons with which they were fully familiar. Following the resolution of the cases where they detected non-equivalent behavior, a group of selected authors would perform the final validation by working through several of their lessons. This method of using knowledgeable TUTOR users is the most meaningful criterion of equivalence, although there may be difficulties in objectively deciding whether the systems are equivalent.

An important issue is the selection of the test sets to be used in Stages Two and Three of testing. In Stage Two, which represents integration testing, the lessons are intended to have no educational value but will be designed to verify that all system commands execute correctly with a wide variety of inputs and that the state of the system is properly modified by each command. Erroneous, as well as correct, lessons would be developed to insure that run-time errors were properly diagnosed. In Stage Three, the lessons to be exercised by members of the PLATO System Group

and the lessons to be exercised by their authors should be selected to contain comprehensive examples of the following TUTOR features:

- Answer judging, including the use of vocabularies and concepts
- Algebraic judging
- Calculations
- Cursor manipulation
- Animation and timed actions
- Display generation.

Each lesson should be documented to summarize the features covered in order to isolate those features not tested by any of the lessons used.

3.4 Performance Testing

Performance testing is intended to measure the response characteristics of the system under a variety of system loads and to assure that the system degrades gracefully when overloaded. For performance testing to be meaningful, the test plan must be repeatable. It must be possible to measure, for example, the behavior of a specified lesson under a known system load before and after system changes are made. If this capability were available and any level of system load could be easily generated, a much more objective evaluation could be made of the quality of service which would be seen by a student. In addition, system loads are required for exercising how the system responds to "unlikely" situations, such as a large number of students requiring lesson material simultaneously or graphic output to many terminals simultaneously.

There are many non-functional features in the quality of the performance of a highly interactive system that are not subject to the normal methods of objective system comparison. The use of the new system, by members of the PLATO System Group and by experienced TUTOR authors will help insure the performance goals for the system are met.

SofTech plans to provide for reproducible performance testing via the use of a minicomputer which can simulate the input from one or more terminals. By the use of previously constructed scenarios, a reproducible multi-terminal environment can be created by the minicomputer for both

integration and performance testing. Such an approach has been used in the development of the Multics time-sharing system and has proved to be extremely valuable. System testing should be possible under much more controlled circumstances than is possible when all inputs occur in a random fashion from a number of terminals. In addition, the minicomputer can record response times and performance-related statistics with minimum changes to the system being tested. The use of the minicomputer to create repeatable system overload conditions should also have a major effect on assuring system reliability.

Section 4

KEY DESIGN DECISIONS

Early in the study several key design decisions were made which bounded the design approaches to be evaluated in detail. These key decisions, which are discussed in more detail below, were:

1. The ability to convert TUTOR programs to the PLATO system from the PDP-11 based system, although desirable, is not an essential requirement.
2. All TUTOR facilities, including those interactive capabilities making a heavy demand on computer resources, will be supported. However, the number of terminals simultaneously making heavy demands for resources must be limited.
3. A TUTOR 60-bit word on the CDC CYBER System will be represented by a block of four bit words (i.e., 64 bits) on the PDP-11 as shown in Figure 4.1. These 64 bits will contain:
   a. Eight characters, each 8 bits with a TUTOR internal 6 bit code right-justified in each PDP-11 character.
   b. One 64-bit floating point value.
   c. One 32-bit integer occupying the first two 16-bit words of the block; the last two words, if actually allocated in storage, are not accessible within a TUTOR program.
4. The authoring of new lessons in TUTOR and the execution of computer-managed instruction programs in other languages are likely uses of the PDP-11. Therefore, it is desirable to perform these tasks in parallel with the presentation TUTOR lessons. However, while running these programs, it is acceptable for the system to support only a very limited number of terminals running TUTOR lessons.
5. The development of the PDP-11 based system can be cost justifiable only if an existing Digital Equipment Corporation operating system is usable with minor modifications.
6. All hardware used to configure the system must be available "off the shelf".

In the following subsection each of these decisions is discussed in more detail.

4.1 Transportability Between TUTOR Systems

SofTech regards transportability of instructional material between computer systems supporting TUTOR as very desirable and a goal to be achieved if at all possible. However, for the success of this project, only the ability to transfer programs from the PLATO System to the PDP-11 System is essential. It is expected that some extensions or modifications to TUTOR, particularly in the areas such as data declarations and the specification of animation, will be required for TUTOR programs to be effectively processed on the PDP-11. Unless modifications are made to the PLATO version of TUTOR, programs using these language extensions will not be transportable from the PDP-11 system to PLATO. However, SofTech decided that if such language extensions should provide the best way to solve a design problem, they should be incorporated into the PDP-11 version of TUTOR.

4.2 Support of All TUTOR Facilities

The decision to support all TUTOR facilities significantly affects the hardware and software architecture of the PDP-11 based system, but is required if one expects to convert existing TUTOR programs to the PDP-11 System without major modifications. If some facilities were not to be supported, significant modifications would be required to the TUTOR language. The most likely facilities not to be supported are those derived from the unique nature of the PLATO environment (see Section 2.2).

The design approach selected was to regard as a critical resource access to the computer resources required to support in an acceptable manner real time interaction with animated displays and/or key echoing requiring lesson material. When the computer resources are not adequate to support all students wishing to enter a portion of a less which uses these facilities, some students will not be able to proceed with their lesson until the resources become available. Since only a small portion of a lesson typically uses these facilities, the approach of waiting for a critical resource appears feasible and should typically produce a delay to the student of only seconds unless the system is seriously overloaded. The details of this design decision are described in more detail in Section 6.4.

An overall conclusion of the study is that the PDP-11 based system is technically feasible and will not require eliminating any TUTOR features. However, the high degree of man-machine interaction implied by some features of the TUTOR language results in a significantly larger hardware cost than is usually associated with a minicomputer system. The software required to support the equivalent execution of the full TUTOR language is complex since it must support very interactive computing in a multiple-terminal environment.

An alternative to the multiple-terminal minicomputer system is a single terminal system supporting only lesson presentation. Such a system would involve significantly less software complexity, since the system would not be required to divide its available resources between several students, while creating the illusion that each student has the computer ready to work for him. With the capabilities of minicomputers rapidly increasing, while total system costs are falling, such an approach should become increasingly attractive during the next three to five years.

4.3 TUTOR Data Representation

A design decision which has impact on many aspects of the PDP-11 system is how a TUTOR word, which is 60-bits on the CDC CYBER, should be represented on the PDP-11. The possible TUTOR word lengths on the PDP-11 are:

1. 32 bits (two PDP-11 words), the minimum length for a PDP-11 floating point value.
2. 64 bits (four PDP-11 words), the length of a PDP-11/45 extended precision floating point value.
3. 80 bits (five PDP-11 words), the storage required to store ten 8-bit characters. Character data on the PDP-11 must be manipulated using 8-bit bytes.

The following factors were particularly important in evaluating the three alternatives:

1. 32-bit floating point values will result in considerably less precision than is available on the CYBER and could result in some subtle problems in converting programs from the PLATO system to the PDP-11 system.

2. Storage allocation for character data will require careful examination prior to conversion if a TUTOR word cannot contain ten characters.

3. 80 bits per TUTOR word is wasteful of storage, since integer and floating point values must be allocated the the same length word, unless the TUTOR language is changed to require explicit data declaration. However, integer and floating point values would use only a portion of the 80 bits since the PDP-11 cannot manipulate floating point values larger than 64 bits or integer values larger than 32 bits.

4. The CDC computer likely to supercede the CYBER System in the PLATO environment will handle 8 bit bytes and will, it is anticipated, have a 64-bit word length.

The choice of the 80-bit word length will reduce the cost of converting lesson material from the PLATO system to the PDP-11 system, but at a significant increase in the storage required to store TUTOR data. Although the 80-bit alternative was seriously considered, the PLATO System Group and SofTech decided that, when all the competing factors were weighed, that the TUTOR word length on the PDP-11 should be 64 bits. Exhibit 4.1 shows how the 64 bits will be used to represent the three types of TUTOR data.

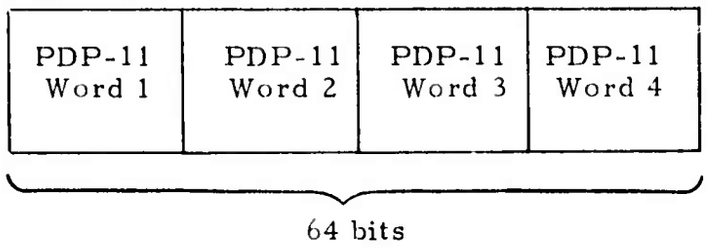
Floating point values will have 55-bit precision on the PDP-11 as compared to 48-bit precision on the CDC CYBER. However, the allowable range of values on the PDP-11 will be from 2^{-128} to 2^{127} as compared to 2^{-1024} to 2^{1023} on the CYBER. Members of the PLATO System Group believe that maintenance of adequate precision is more important in minimizing conversions problems than maintaining such a large range of values. Some TUTOR authors appear to use the high precision of the CYBER as a substitute for fully understanding the computational properties of their algorithms.

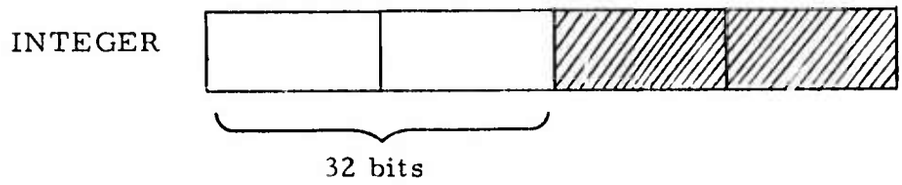
A character will be represented by an 8-bit byte containing 6-bit PLATO internal code right-justified in the byte. Capitals, subscripts and superscripts will be represented by a two-character sequence, equivalent to that used on the PLATO System. The translation of all input characters in the internal representation used on the PLATO System will greatly reduce conversion problems and will not significantly affect the software complexity.

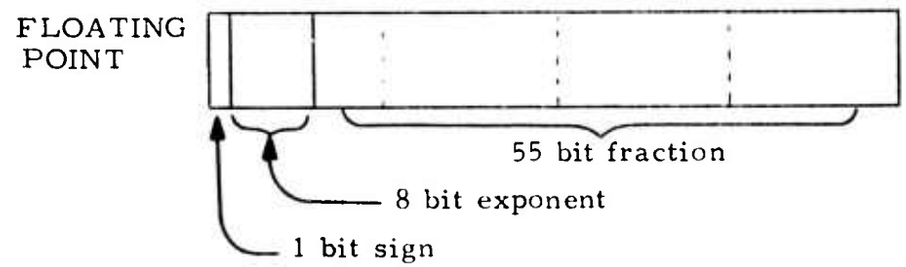
Exhibit 4.1

TUTOR DATA REPRESENTATION

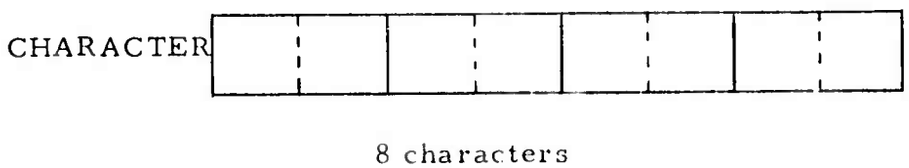
TUTOR WORD







Exponent Range: 2^{-128} to 2^{127}



4.4 Simultaneous Execution of non-TUTOR Programs

The PDP-11 system could be designed to only support the execution of TUTOR programs, or a more general approach could be selected which would allow the simultaneous execution of programs written in languages other than TUTOR. Only the more general approach would allow the authoring of lessons during lesson presentation. Another reason favoring the more general approach is the likely interest in supporting computer-managed instruction, which is likely to require the execution of programs written in a language other than TUTOR.

The early stages of the design effort indicated that providing a highly interactive system was incompatible with other simultaneous uses of the minicomputer-based system. Therefore, SofTech assumed that use of the system would be divided into time periods of heavy student use and time periods during which new material was developed on programs executed for computer-managed instruction. During the period of heavy student use, the system would operate in a mode optimized to support the interactive execution of TUTOR programs. At other times, the system should be capable of executing programs written in other languages while supporting, if desired, a very small number of terminals executing TUTOR lessons. In this way an author could input and execute new lessons during a single session on a terminal. The priority provided to each of the simultaneous functions being performed by the system should be easily modifiable, depending upon an installation's policies.

4.5 Use of DEC-Supplied Operating System

The cost of developing and maintaining an operating system for the PDP-11 to support this application is, in SofTech's opinion, sufficiently high that if such an operating system must be developed the overall system will not be cost-effective. The cost of developing and documenting a PDP-11 operating system capable of supporting the highly interactive requirements of TUTOR programs, while meeting the above requirements for supporting the execution of non-TUTOR programs, could easily be several hundred thousand dollars. In addition, the use of a DEC operating system already in widespread use should significantly reduce the maintenance costs for the overall system.

The only DEC operating system that could possibly serve as the base for this design is RSX-11D, a resource-sharing executive designed for use in real-time applications. The selection of RSX-11D restricts the choice of PDP-11 processor to either the PDP-11/45 or the PDP-11/40 with the Extended Instruction Set.*

4.6 Use of Commercially Available Hardware

SofTech has configured the system using commercially available hardware in order to avoid speculating on hardware developments that may, or may not, occur. However, hardware developments are likely to occur during the next two or three years could significantly increase the capabilities of the system while reducing the costs.

*This report was completed prior to the announcement of the PDP 11/70 (Section 9).

Section 5

TUTOR LESSON STRUCTURE

The PLATO System design depends heavily on the access and transfer rate characteristics of the electronic swapping memory (Extended Core Storage or ECS) available on the CDC CYBER series. The PLATO memory management strategy, which involves moving units of a TUTOR lesson into central memory for execution as they are referenced, is feasible if a unit can be obtained from ECS and made ready for execution in only a few microseconds. The transfer rate from ECS to central memory is approximately ten million 60-bit words per second. All data that might conceivably be used by a lesson is transferred to central memory before lesson execution begins.

The strategy used for manipulating program and data will be substantially different when disk or drum swapping memory is used. Such memories are two orders of magnitude slower in transfer rate than ECS and three to four orders of magnitude slower in access time. The following table compares the highest performance fixed-head disk for the PDP-11 with CDC Extended Core Storage.

	CDC ECS	DEC RJS04 Fixed Head Disk
Access Time (microseconds)	5	9000 (average)
Transfer Rate (bits/second)	600×10^6	4×10^6

If all of the programs required to execute TUTOR lessons, as well as the lessons and data required to handle each student, could be kept in the core memory of the PDP-11 while a terminal interaction was being processed, the access and transfer rate limitations of the fixed-head disk would not be important. Since the cost of this approach is exorbitant and not technically feasible with commercially available hardware, the design of the PDP-11 based system will be successful only if it can significantly reduce the amount of information transferred in the PLATO

system between central memory and secondary storage. The critical resources in the PDP-11 system are core storage and traffic between central memory and the fixed head disk; a more detailed analysis in Section 6 shows that a PDP-11/45 CPU will be significantly under-utilized at the time when the other system resources are fully utilized. SofTech, with extensive assistance from the PLATO Systems Group, has analyzed the use of TUTOR lessons and data in the PLATO system and has developed the strategies described later in this section for executing TUTOR lessons on a system without electronic swapping memory.

5.1 Properties of TUTOR Lessons

A brief review of the organization of a TUTOR lesson is useful for understanding the implications of the strategies to be subsequently described. A TUTOR lesson typically consists of several main units, i.e., a main unit sequence (MUS), which can call upon attached units, as shown in Exhibit 5.1. Execution of a lesson begins with a specified main unit, and control will be transferred upon the completion of that main unit to another main unit, usually selected as the result of evaluating a conditional expression. Control will never return to the main unit just completed unless an explicit transfer is made to it. The use of an attached unit is similar to a subroutine call and control will return to the main unit which invoked the attached unit. Arguments can be passed to an attached unit but not between main units.

A lesson may consist of up to seven main unit sequences, one of which is initially executed when the lesson is started. The other six main units or auxiliary main unit sequences (AMUS) are involved via one of six function keys or by a TUTOR command. Control returns to the main unit executing when the first new sequence was invoked unless another main unit is explicitly specified. Such optional sequences allow "help" material to be developed which the student can request by pressing the HELP function key. Such sequences can also be used to provide review material, a desk calculator, a dictionary of terms, tables of data, or any type of material that is related to the lesson being studied but which the author does not require all students to study. The use of a help sequence and a desk calculator is illustrated in Exhibit 5.2; note that one unit is attached from both the MUS and an AMUS.

Four points about the structure of a TUTOR lesson are important in designing a strategy for accessing units during the execution of a lesson.

1. A TUTOR lesson, including both the MUS and all AMUS, is a single compilation unit and the lesson structure is completely specified at compile time.

2. All references from one unit to another unit appear explicitly in the program text. The language does not allow implicit references via label or procedure variables.

3. Main units divide the lesson into segments that are quite independent of other segments, since arguments cannot be passed between main units and since control does not automatically return to the invoking main unit. A significant fraction of main units do not use a conditional branch to select the next main unit executed and, therefore, the flow to the next main unit is sequential.

4. Transfer from the MUS to an AMUS takes place infrequently and constitutes a natural "break" in the flow of the lesson. However as much as half of the lesson material may be found in these subsidiary sequences.

5.2 Strategy for Accessing TUTOR Lessons

The PLATO memory management system is quite simple and relies upon the low access time and high transfer rate of ECS memory. The system loads all units of a lesson into ECS from disk when the lesson is first accessed. A central memory buffer contains the most recently used units at all times and, if a referenced unit is not in this buffer, it replaces one or more units in buffers. No attempt is made to predict what units will be needed in order to load them in advance of being referenced.

On the PDP-11 system, with its much longer disk access time and slower transfer rate, more units must be accessed and loaded with one disk transfer and these units should have a high probability of being executed. If adequate core storage were available, one strategy would be to load all the units in the lesson the student was executing; only one disk access would be required to obtain all the material. However, the transfer would be lengthy since the equivalent of 3,000-4,000 60-bit words, the average length of a TUTOR lesson on the CYBER, would be transferred.

Exhibit 5.1

BASIC STRUCTURE OF A TUTOR LESSON

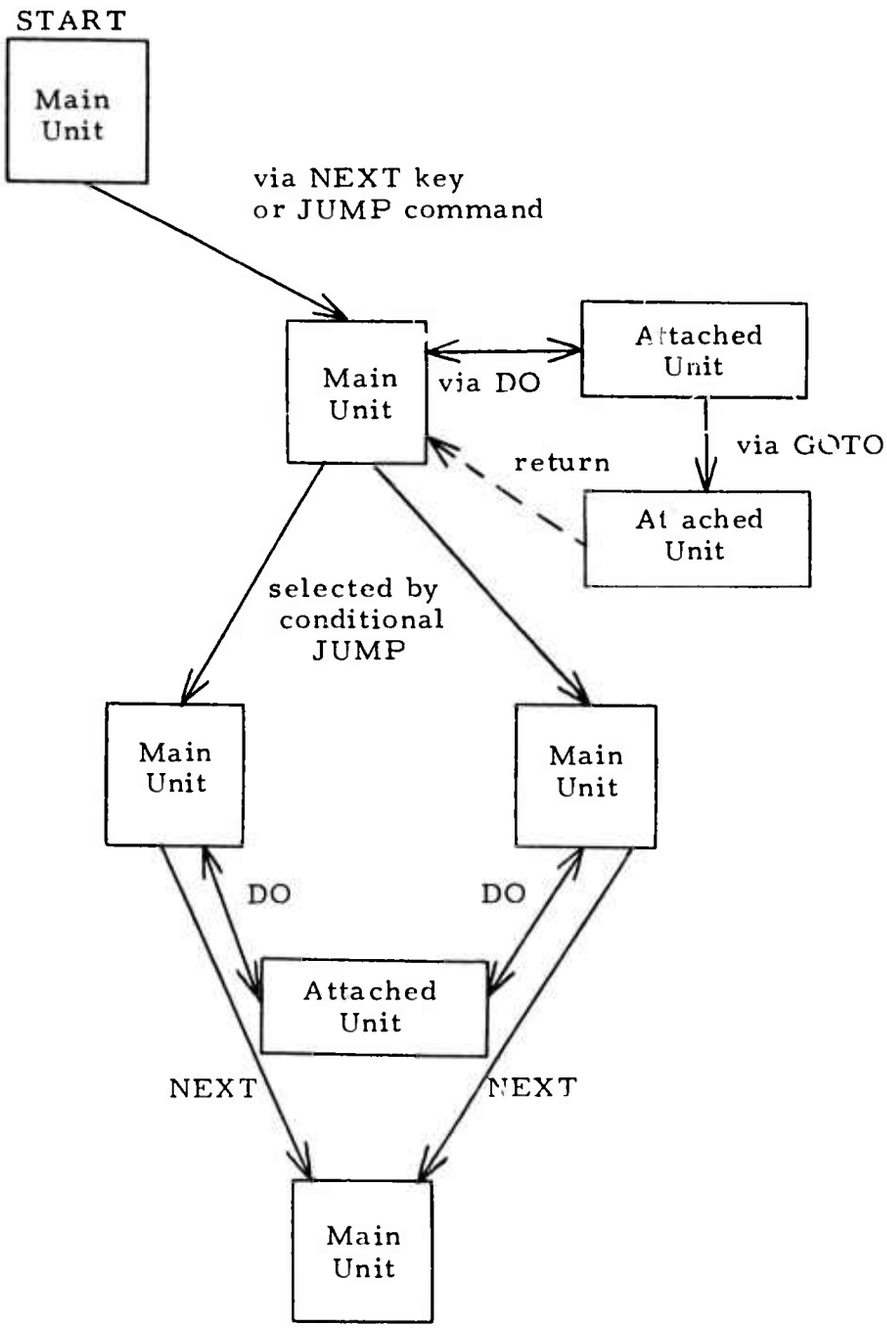
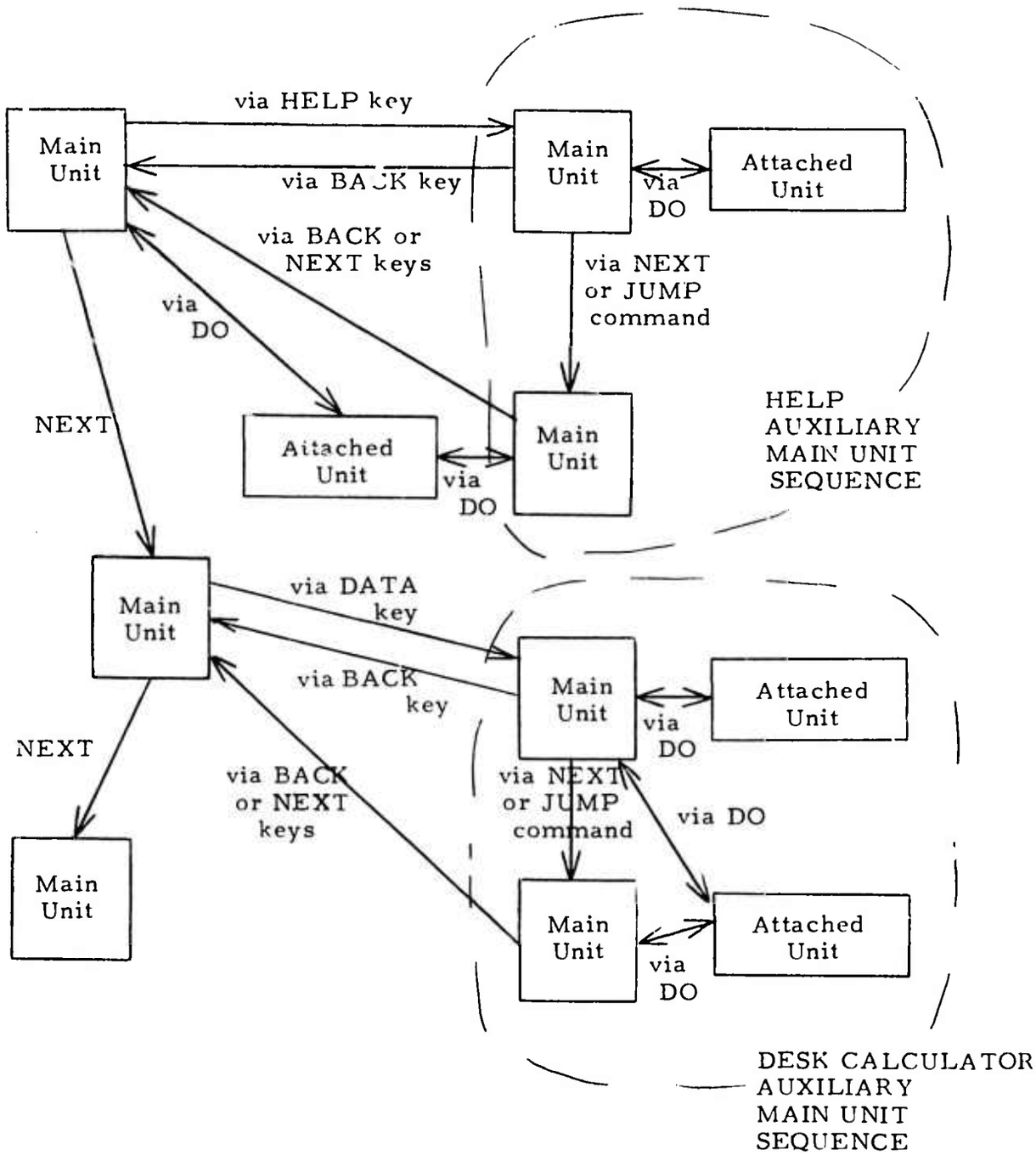


Exhibit 5.2

USE OF AUXILIARY MAIN UNIT SEQUENCES



Since a significant fraction of the lesson consists of auxiliary main units, which are seldom invoked, an alternative strategy is to store the lesson on disk in the form of segments, one for the MUS and one for each AMUS; this division is illustrated in Exhibit 5.3. When a main unit in a different sequence is invoked, the segment containing all units in that sequence would be loaded; units attached in several main unit sequences will be duplicated in each segment. This strategy requires one disk access for every segment loaded, but the amount of information transferred and the amount of core storage required is less than if the entire lesson were to be loaded. If the amount of central memory space is not adequate for executing a segment representing a complete main unit sequence, the segment would contain one or more main units plus all the units it attaches as shown in Exhibit 5.4. Units referenced in more than one segment will be duplicated so that as few references as possible are made to units in other segments.

The following strategy for loading and executing TUTOR lessons takes full advantage of the structure of a TUTOR lesson to reduce the number of disk accesses and to execute the lesson using less central memory space than would be required by the entire lesson.

1. Compile each TUTOR unit into a machine-readable format that will allow the Segment Builder to combine it with other units to create segments. TUTOR programs will not be recompiled each time the lesson is used, as has been the strategy at PLATO.
2. Determine the central memory space allocated for executing TUTOR lesson material for an individual student. The space allocation must be known at the time Step 3 below is performed.
3. Combine the units of a main unit sequence to produce the largest possible segment which will fit within the available central memory space. The segment must contain all units attached by any one of the main units.
4. If one main unit and all its attached units requires more space than is allocated for TUTOR lesson material, the main unit must be subdivided by the lesson author into two or more main units. Each of the new main units and its attached units must fit in the central memory allocation.

The strategy outlined above divides the lesson into sections, each of which will fit into the available central memory space, in a manner that reduces the probability that when a unit is referenced it is not in central memory.

The segments and blocks stored on disk can be loaded with one disk access and disk transfer. A Lesson Description Table is built by the Segment Builder and contains:

1. An entry which will be used to associate each segment name with the disk location where it is stored.
2. An entry associating the name of each main unit referenced from a unit in another segment with the segment and the position within the segment where that unit is located.
3. A description of whether the segment uses any of the COMMON and/or extra storage data used in other parts of the lesson (see Section 5.3).
4. The disk location of COMMON, extra storage and vocabulary data, and Charset and Micro tables used by the lesson.

The Lesson Description Table must be in central memory throughout lesson execution, since it will be used to load all segments and data from swapping memory and to handle all transfers from main units in one segment to those in another segment.

5.3 Strategy for Accessing TUTOR Data

The TUTOR data required to process a student interaction can require considerably more central memory space than the TUTOR lesson material. As Section 2.2 indicates, more than 4000 60 bit words of data may be required to handle an interaction which typically executes only 200 words of lesson material. However, many interactions will actually use only a small portion of the data which is potentially accessible. Therefore, in order to reduce unnecessary disk accesses and disk transfers, only data which is expected to be used in processing the interaction should be loaded from swapping storage.

The following strategy for managing TUTOR data on the PDP-11 assumes that the lesson material will be stored on disk as segments, in the manner described in Section 5.2:

Exhibit 5. 3

DIVISION OF A LESSON INTO THREE SEGMENTS

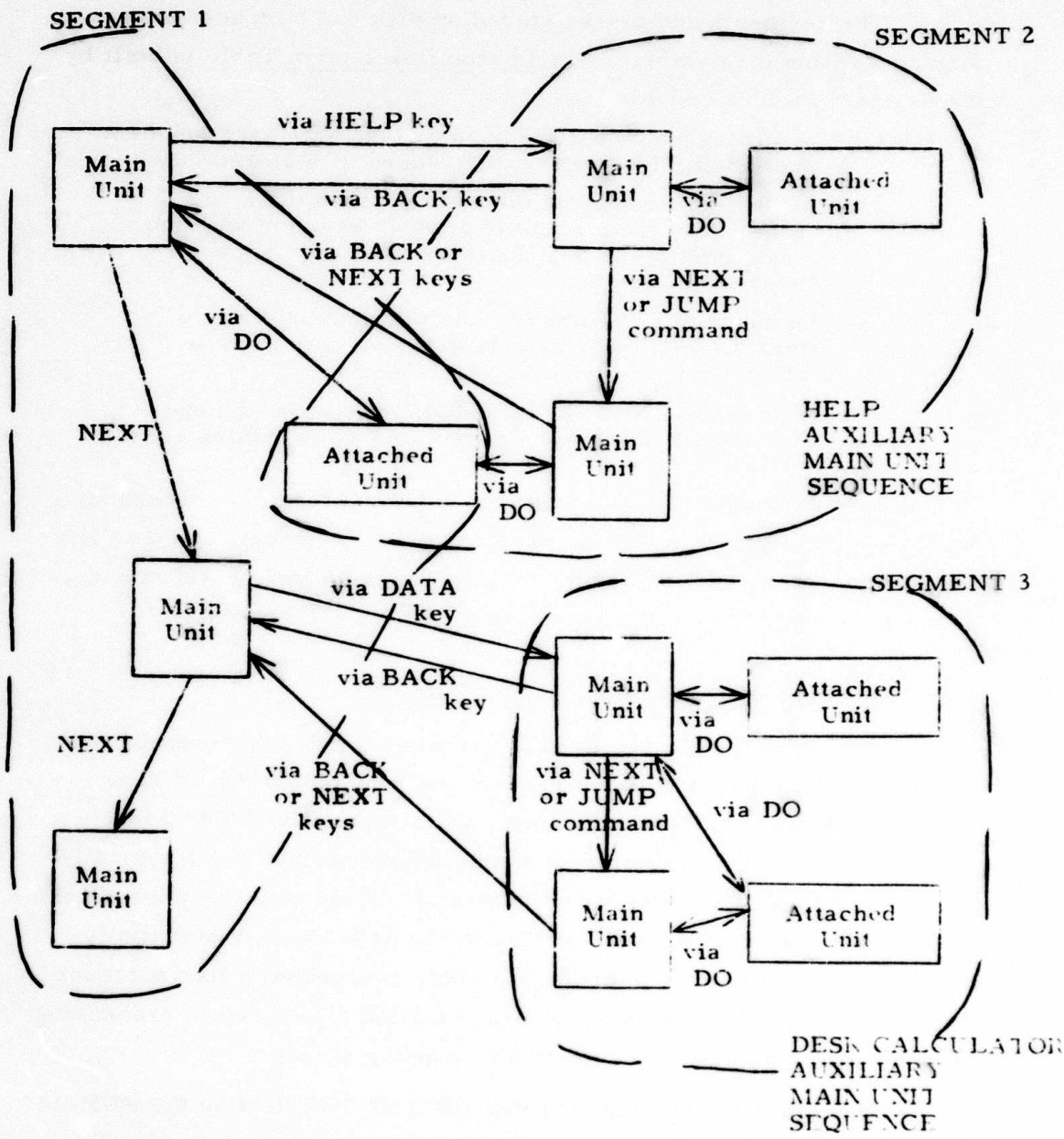
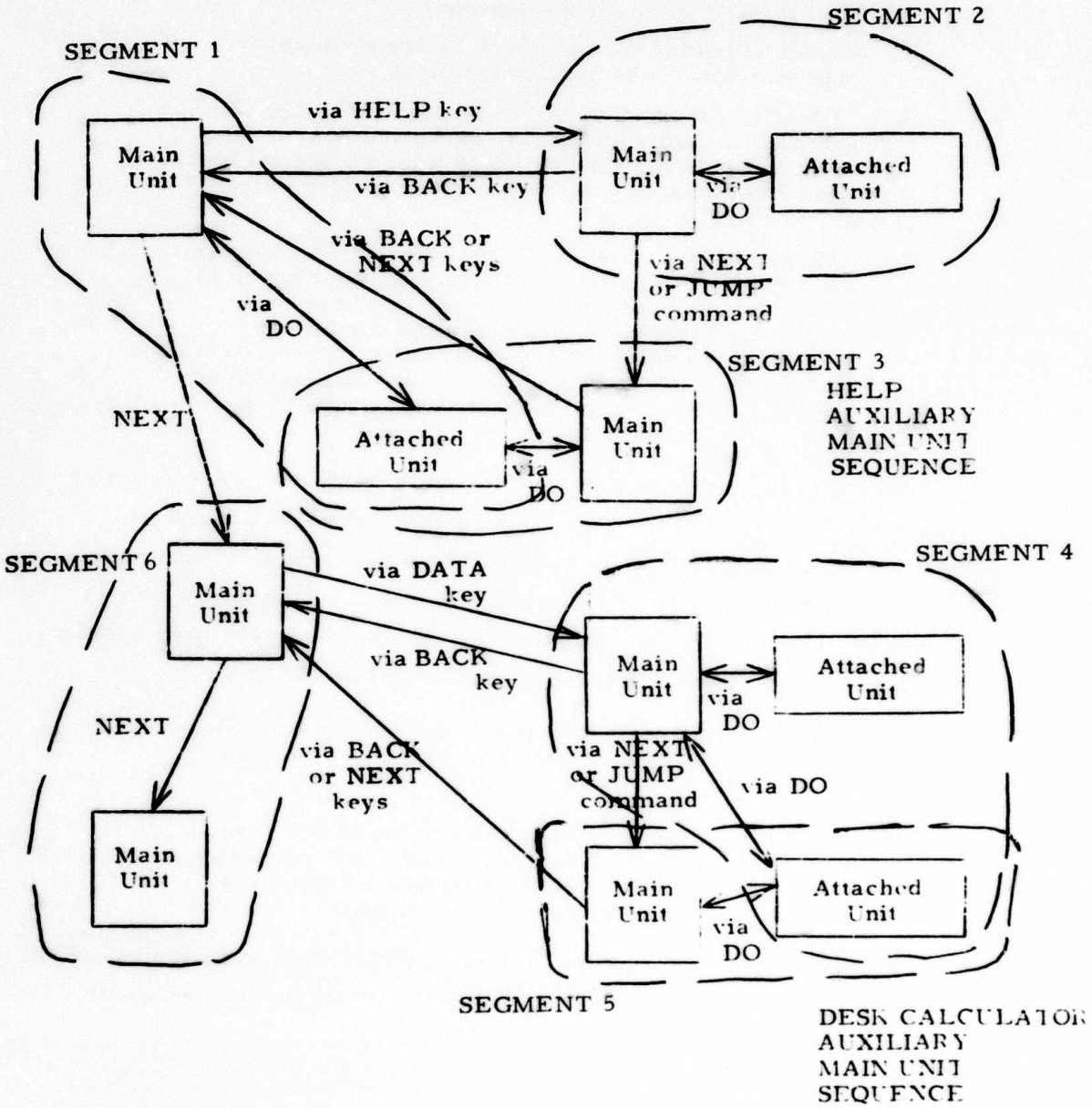


Exhibit 5.4

DIVISION OF A LESSON INTO SIX SEGMENTS



1. TUTOR student variables for each active terminal are kept in central memory at all times and written to long-term disk storage only when the student terminates his use of the system.

2. Terminal/status data for each active terminal is kept in central memory at all times.

3. COMMON data and/or extra storage data will be transferred into central memory from swapping storage when a segment is loaded if the lesson material being loaded references the Common data and/or extra storage data.

4. COMMON data and extra storage data will be written back to swapping storage when the processing of an interaction is complete only if one or more data values were altered while executing the lesson material. COMMON data is written back to long-term disk storage from swapping storage when no active terminals are running the lessons of which it is a part.

Two copies of a lesson COMMON cannot be in central memory at the same time. COMMON data is a shared resource that provides for sharing lesson data among all terminals using that lesson. Access to the data in swapping storage must not be allowed if a copy is already in central memory since the central memory version may have been updated, but the updated copy has not been returned to swapping storage.

5. A vocabulary for response analysis will be loaded from swapping memory into central memory at the time it is needed.

6. A micro table for key echoing must be kept in central memory at all times, since the response time required for key echoing will not permit a disk access and transfer upon demand.

The estimated size of each of these TUTOR data types is provided below as well as a description of how the strategy outlined above might be implemented.

1. TUTOR student variables (600 PDP-11 words -- 150 60 bit values) will, with high probability, be referenced in every segment. They must be preserved in long term storage between student sessions.

2. Terminal and lesson status data (approximately 500 PDP-11 words, which will provide, in a densely packed form, the same information as is stored in 250 60-bit PLATO words) must be available throughout a student session, but only a small fraction of the data needs to be saved between student sessions. It will be used in every segment.

3. COMMON data (up to 6000 PDP-11 words -- 1500 60 bit values) will be resident on swapping memory while one or more terminals is using a lesson and will be loaded into central memory only while processing those segments where it is referenced. The compiler output must indicate, for each unit, whether it references COMMON so that the Lesson Description Table will reflect whether any unit in a segment references COMMON. If one or more units in a segment use COMMON, the COMMON data will be loaded when the segment is loaded. The interpreter will mark the COMMON data as modified if one or more values are altered, so that the modified values will be written to disk for use when the next interaction is processed. If the author so specifies, COMMON data must be saved between uses of the lesson.

4. Extra Storage (up to 6000 PDP-11 words - 1500 60-bit words) will be treated in a manner very similar to COMMON data. However, this type of data is not saved between uses of the lesson.

Less than 25 percent of all existing TUTOR lessons use COMMON data and only a very small percentage use extra storage. There is a PLATO restriction that the total of COMMON and extra storage can not exceed 1500 60-bit words, or 6000 PDP-11 words. In nearly all lessons, the total size of COMMON and extra storage data that must be simultaneously resident in main memory is less than 1000 60-bit words. Therefore, if desirable, a lower threshold than a total of 1500 60-bit words could be established for the PDP-11 based system.

5. Vocabularies for response analysis (up to about 10,000 PDP-11 words -- 2,000 - 3,000 60-bit words) can be loaded into central memory only when they are needed to judge a response. Responses of sufficient complexity to require a vocabularly for judging will be input to the system at a very slow rate, perhaps one response every few minutes per terminal and a slight delay while the response is being judged will not upset the flow of the lesson.

6. Micro tables for key echoing (up to 1280 PDP-11 words or 256 60-bit words) must be kept in central memory at all times in order to perform key echoing without a delay which will make touch typing difficult.

A small amount of other data, (e.g., Charset tables for loading a new character set into the PLATO terminal) can be associated with a lesson but, if included, this data is not accessed frequently nor required for a long period of time. This type of additional data is not important for deriving an estimate of the amount of central memory space required per terminal executing a TUTOR lesson or the volume of information which must be transferred from disk storage to central memory.

The resulting central memory requirements for executing a TUTOR lesson when the maximum amount of data allowed is being used would appear to be:

Student Variables and Terminal/ Status Data	1100 words
COMMON Data and Extra Storage Data	$\leq$ 6000 words
Vocabulary Data	10000 words
Micro Table	1280 words
	18380 DDP-11 words

Vocabulary data is required only while evaluating a response, and little, if any, of the other lesson data is required during that process. Therefore, one approach to reducing the amount of data in central memory, assuming the PDP-11 will perform multi-programming, is for the response analyzer program to operate using a queue of requests from a TUTOR lesson. Each request would specify the student response to be evaluated as well as the name of the vocabulary data to be used for the evaluation. Since response evaluation is a very rapid process when the algorithms (7) developed by Tenczar are used, queuing the requests for processing should not introduce a delay that is annoying to the student. Since a binary search of the vocabulary data is performed in the response analysis algorithm the amount of data to be loaded can be halved at

the expense of one additional disk access. Therefore, all subsequent central memory estimates will assume a 5,000 word buffer will be required.

If the above approach is adopted, the data per lesson being executed in central memory can be reduced to a maximum of approximately 8500 PDP-11 words, as long as one buffer of 5,000 words exists as part of the response evaluation program. Since most lessons will require much less than 8500 words of data, 10,000 PDP-11 words appear adequate for containing both the TUTOR lesson material and the associated data. If the maximum amount of data is being used, only a small amount of lesson material will be in central memory and system response will suffer if frequent references are made to main units outside the segment currently loaded. However, for the majority of lessons which will require only 1100 words of data (student variables and terminal status data) a large fraction of the lesson material will be able to be loaded into central memory.

5.4 Strategy for Interpreter Implementation

The PLATO TUTOR interpreter is a large program requiring approximately 12,000 60 bit words, which is not unexpected since the author subset of TUTOR contains approximately 200 commands. The PDP-11 based system will be unfeasible if the entire interpreter must be core resident. As indicated in Appendix B, SofTech estimates that a CDC 60-bit instruction is roughly equivalent to 5 to 6 PDP-11 16-bit instructions, assuming the CDC code is written in tight assembly language and the PDP-11 code is developed using a high-level systems implementation language with an optimizing compiler. On this basis, 72,000 PDP-11 words would be required to store the interpreter, but the maximum core configuration supported on a PDP-11/45 is 124,000 words.

Since few, if any, lessons use more than half of the TUTOR commands, a significant fraction of the code in the interpreter is not used while running a lesson and an even smaller fraction would be used to process a segment. Statistics gathered on the PLATO system show

that 78 percent of the interpreter execution time is spent in ten percent of the commands (20 commands). The central memory required to execute these particular 20 commands is about 1500 60-bit words of command execution code, plus about 2,000 60-bit words for the overall framework of the TUTOR interpreter. Each lesson has its own set of commands and needs only a fraction of the 10,000 60-bit words of command execution code as indicated by the few typical lessons listed below:

Subject Area	Commands Used	Length of Command Execution Code (60-bit words)
First-year Physics	79	7053
First-year Chemistry	54	5165
Second-year Chemistry	47	5201
First-year Russian	54	3401

The strategy selected for the PDP-11 system is to keep core resident the framework of the TUTOR interpreter plus the code required to execute the 20 most frequently executed commands. All other command execution code required to process the commands used by a segment could be stored on swapping memory as part of the segment. This approach allows both the TUTOR lesson material and the command execution code for infrequently used commands to be obtained from swapping memory with one disk access and transfer.

The above approach has the disadvantage that since a few infrequently used commands require a significant amount of code, this code will unnecessarily be duplicated in swapping memory. Some of these commands are:

Command	Approximate Size (60-bit words)
Graph Plotting Package	1200
Setup new character set	250
Run-time Arithmetic Expression Compiler	2000
Student Data Logging	400

A small delay in executing these commands is acceptable since they are not likely to be included within the highly interactive parts of a lesson. Therefore, loading the command execution code for these commands upon demand is a feasible alternative should the amount of code duplication in swapping memory be excessive. In Section 7 the central memory requirements for command execution code are discussed in more detail.

The strategies described in this section will significantly reduce the number of transfers from swapping memory and will, in SofTech's opinion, make it feasible to develop a highly interactive system using a high speed disk for swapping memory, rather than the electronic swapping memory that is basic to the design of the PLATO system.

Section 6

SYSTEMS ARCHITECTURE

In this section the following key design issues involved in the development of the PDP-11-based TUTOR system are discussed:

1. What important limitations does the use of the Digital Equipment Corporation RSX-11D operating system place on the design?
2. What are the most critical system resources, and is there an approach to effectively utilizing these resources that will guarantee the system's feasibility?
3. How should the system handle highly interactive lessons (i.e. those lessons that involve real-time interaction with animated displays or the execution of lesson material during key echoing)?
4. What problems are introduced if the system is required to support services other than the presentation of TUTOR lessons (i.e., development of TUTOR lessons or the execution of programs for Computer Managed Instruction)?

The answers to these questions are discussed in the following subsections but, in summary, the study showed that:

1. RSX-11D, the DEC operating system designed for real-time applications, significantly complicates the management of core memory, but provides the efficient multiprogramming facilities required to implement the system.
2. The most critical resources are the system facilities required to support the transfer of information to and from swapping memory. The criticality of these resources depends on the amount of central memory which is available on the PDP-11.
3. Only one terminal at a time will be able to run a portion of a lesson requiring real-time interaction with animated displays or the execution of lesson material during key-echoing. The author will be required to specify the relevant part of the lesson by enclosing it within directives which will be added to the PDP-11 version of the TUTOR language.
4. Services other than lesson presentation, such as executing programs for computer-managed instruction, will be supported, but when they are in use the number of terminals executing TUTOR lessons will be significantly reduced.

The system design did not reveal any reasons why the PDP-11 based system was technically infeasible. The study indicates, however, that the hardware investment required will be larger than originally anticipated if the system is to support 24-32 active terminals. The proposed hardware configuration is described in detail in Section 7.3.

6.1 Key PDP-11 Hardware Characteristics

Some characteristics of the PDP-11 series of computers are of particular importance to this project and are summarized in this subsection; a more comprehensive description is provided in Digital Equipment Corporation documentation (8, 9). The PDP 11/45 is the basis of this design since it is the only member of the PDP-11 line which is available with a processor that supports operations on both 64-bit floating values and 32-bit integer values. The only other alternative is using an 11/40, which will require software simulation of 64-bit floating point arithmetic using the hardware-supported 32-bit floating point operations, and software simulation of some 32-bit integer operations. The difference in cost between the two processors is less than $12,000 and increased software costs associated with using the less capable hardware facilities will be at least as much as the difference. Therefore, the PDP 11/45 has been selected for use in the system. At the time this report was being prepared the PDP 11/70 had not been officially announced and a full description of its capabilities were not available.

A comparison of speed of the PDP 11/45 versus the PLATO CDC CYBER 73 using hand comparisons based on published instruction timings is provided in Appendix B. The results indicate that the 11/45 is approximately 3-5 times slower than the CYBER 73. Therefore, the 11/45 should not be CPU bound when delivering lessons to 20-30 terminals since a one processor CYBER 73 would be capable of delivering lessons to well over 150 active terminals if no authoring of new lessons is simultaneously taking place.

Up to 124 thousand (124K) 16-bit words of memory can be attached to an 11/45 if the Memory Management Unit (MMU) is included, which enables more than 32K words to be addressed. Up to 64K words can be

addressed with the MMU at one time, of which 32K words is instruction space and 32K words is data space, which cannot be used for execution. The instruction (or data) space need not consist of 32K words of contiguous physical memory but can represent a 32K word virtual memory consisting of eight 4K word pages. The mapping between virtual and physical memory is done using eight Page Address Registers for the instruction space and eight registers for the data space. The length of a page can be set to less than 4K words via a Page Description Register, but when a shorter length is given, the physical memory locations within the 4K word page but beyond the bound specified are not accessible. Therefore, at any time not more than 32 K words of instructions and 32K words of data can be referenced, and each 64K virtual memory cannot be divided into more than sixteen blocks of physical memory. Some of the limitations of this virtual memory approach will become more obvious when the organization of memory under the RSX-11D operating system is discussed in detail in Section 7.2.

A variety of disk storage devices for the PDP-11 are available ranging from a high performance fixed head disk to a large capacity disk storage system with features similar to those of an IBM 3330 model. The following are the characteristics of the models most relevant to this study:

Model	Capacity ($x10^6$ words)	Average Access (milliseconds)	Transfer Rate (μsec/word)
RJS04 (fixed head)	.5/disk	9	4
RJP04 (similar to IBM 3330)	44/drive	36	2.5

The RJS04 fixed head disk is the logical choice for the swapping device which will hold all lesson material and data for active students. The RJP04 (or the equivalent supplied by an independent peripheral manufacturer) would provide the capacity needed for the long-term storage of lessons and student data.

A two processor PDP-11 system may be developed using either a UNIBUS Link or a UNIBUS Window. The UNIBUS Link allows memory-to-memory transfers through the UNIBUS of one system to the UNIBUS

of the other system. The UNIBUS Window allows one processor to access the other's memory address space as if that address space were part of its own address space. The location in the originator's address space is fixed, but the address in the target's memory can be relocated over the entire address space.

6.2 Key RSX-11D Operating System Characteristics

RSX-11D is a multi-tasking operating system for the PDP 11/45 which has been designed to support real-time applications (10, 11) and is the only DEC-supported operating system suitable for developing an interactive system with many of the characteristics of PLATO.

Work is performed under RSX-11D by scheduling tasks, which may have differing priorities, to execute in defined areas of memory called partitions. A task is the basic RSX-11D program unit, and a program consists of one or more tasks. A task can execute by itself or in conjunction with other tasks or other partitions. The time for RSX-11D to switch from one task to another, if both are resident in memory, is approximately 1.5 milliseconds.

The environment in which a task executes is a virtual PDP 11/45 having 32K words of virtual address space which can contain both instructions or data. The concept of a 32K instruction space and a 32K data space is not currently supported by RSX-11D. Each 4K word block is addressed via one of the eight Page Address Registers (PAR). The virtual memory of a task can be protected by the hardware from other tasks executing in the system since the task's memory can only be addressed via the PAR's for the task. The virtual address spaces of two (or more) tasks can be made to correspond to the same physical memory space to allow the use of shared libraries and to provide for a data space (the Common Area) to be used for intertask communications.

Partitions contain executing tasks and tasks that are permanently resident in memory regardless of whether or not they are executing. All memory, except that required for the RSX-11D Executive and system subroutines (approximately 24K words) is part of some partition. At system-generation time the size of each partition is fixed ($\leq$ 32K words) as well as whether the partition is user-controlled or system-controlled.

A user-controlled partition can contain one task at a time and would usually contain a resident real-time task. A system-controlled partition can contain one or more tasks at a time with the number depending upon whether there is a large-enough, physically contiguous area of memory remaining within the partition for the additional task to be loaded. Tasks are not moved in memory to make room for additional tasks. A task can be fixed in core memory so that it is never swapped out to make room for a higher priority task. A task, such as the TUTOR compiler, can be made "checkpointable" so that when a higher priority task requests the partition, the executing task can be pre-empted, swapped out to disk, and later swapped in and restored to execution at the point where it was previously pre-empted. Since an RSX-11D task is contiguous in physical memory, and its checkpoint disk area is established at system generation time, the task can be swapped with a single seek and write, which minimized the initiation delay of the pre-empting task.

A task may be composed of an overlay structure consisting of a root segment and a number of branch segments. The tree structure overlay scheme supports both the loading of segments when requested and the automatic loading of segments when a transfer of control is made to an entry point in a segment which is not in core. An overlay segment can be brought into core with a single disk access.

A key aspect during system design was to decide how to use the facilities provided by RSX-11D in the most effective manner. Therefore, the design effort involved dividing the job of presenting a TUTOR lesson into several cooperating and communicating RSX-11D tasks.

6.3 Critical System Resources

The heavy dependence in the design of the PLATO System on electronic swapping memory (ECS memory) implies that unsatisfactory performance would result if disk memory was substituted for ECS memory. Although such a problem is unavoidable if the PLATO strategy is adopted for transferring data between main memory and secondary storage, the strategies described in Section 5 were designed to reduce the disk usage to a tolerable level. In this section a detailed estimate is provided of the expected disk usage if these strategies are used.

The average number of disk accesses and the disk transfer time required to process a student interaction can be estimated using Exhibit 6.1. The following data is assumed to be kept in core at all times:

a) TUTOR student data and terminal/lesson status data

b) Lesson Description Table

The Lesson Description Table is assumed to contain all the information needed to read a segment or other data without first reading any type of file directory from swapping memory.

The estimate in Exhibit 6.1 of the lesson length of 2000 words per segment indicates that a typical lesson would be divided into at least 8-12 segments assuming some frequently used units will have to be copied into several segments. This size segment should, in most lessons, include more than one main unit and, therefore, the probability of needing a second (or third) segment should be quite low.

The size of the command execution code to be loaded as part of a segment is the biggest unknown and was estimated using the PLATO data that 4000 60-bit words (20,000-24,000 PDP-11 words) of command execution code is typically required, in addition to the 1500 60-bit words for the 20 most frequently used commands. 7,500 - 9,000 PDP-11 words of the command execution code is assumed to be loaded only upon demand, thereby leaving 12,500 - 15,000 PDP-11 words to be divided among the 8-12 segments. Some duplication of command execution code will occur in each segment. The estimate of 3,000 PDP-11 words of command execution code per segment, therefore, appears conservative.

Exhibit 6. 1

ESTIMATES OF INFORMATION TO BE TRANSFERRED
BETWEEN CENTRAL MEMORY AND SWAPPING MEMORY
PER INTERACTION

Reason for Disk Access	Words to Transfer	Probability Required per Interaction
Load segment containing: TUTOR lesson material (2000) Command execution code (3000)	5000	100%
Load Second Segment	5000	50%
Load Third Segment	5000	20%
Load Common Data	3000	25%
Load Extra Storage Data	3000	10%
Load Vocabulary Data	6000	20%
Load infrequently-used Command execution code	8000	25%
Write Common Data	3000	15%
Write Extra Storage Data	3000	10%

If the basis of making the estimate is valid, then per interaction requiring lesson material (i.e., approximately every 8 seconds per terminal on the PLATO System), the PDP-11 system will make an average of 2.75 disk accesses and will transfer 13,500 words. This will require 79 milliseconds using a fixed head disk having a 9 millisecond access time and with a transfer rate of one word per 4 microseconds. Therefore, the disk is estimated to be in use about 2.5 seconds out of 8 seconds to support 32 terminals, assuming no other disk I/O is performed by the operating system. On the basis of this estimate the 11/45 system should be able to support the design goal of 24-32 terminals with a reasonable load on the disk.

It is interesting to repeat the estimate with the following less optimistic assumptions:

1. The student status variables, the terminal/lesson status information, and Lesson Description Table must be loaded from disk in order to process each interaction and returned to disk.
   Add: 2 accesses, 2800 words transferred
2. Second segment (80% probability instead of 50%)
3. Third segment (40% probability instead of 20%)
4. Command execution code per segment is 5000 words instead of 3000.
5. Infrequently used command execution code (50% probability instead of 25%).

These changes yield the extremely conservative estimate that while processing an interaction the PDP-11 system will make an average of 4.75 disk accesses and will transfer 25,000 words, requiring 143 milliseconds. Therefore, the disk is estimated to be running about 5.5 seconds out of every 8 seconds to support 32 terminals.

The above results indicate that the access and transfer speed of the fixed head disk would become a problem only if there was not sufficient central memory space to store the student variables, the terminal/lesson information, and the Lesson Description Table for each user. If central memory space was sufficiently limited to force a significantly smaller segment size, the capacity of the disk to access and transfer information could also become a problem.

6.4 Executing Highly Interactive Lessons

A large majority of existing TUTOR lessons do not exploit the capabilities of the PLATO system to execute highly interactive lessons. However, those few lessons that do present a particular challenge for the PDP-11 based system. Two simultaneous users making heavy demands for system resources pose a much larger problem on a 32 terminal system than they would on a 320 terminal system.

Two examples may help illustrate the type of interactions possible on the PLATO System:

1. A cursor can be moved rapidly in 8 directions by typing the appropriate key continuously. The key typed is not plotted, but rather a "+" is plotted at the new calculated cursor position.
2. The student watches a changing display until he recognizes a pattern and reacts by typing a character indicating he has seen enough to enter an answer. His score is based upon how little of the pattern he has to see before giving a correct answer. Note that we are interested in how much of the pattern has been displayed, not how much may have been calculated.

In both cases the timing between student input and system output is important. The first case, in which a calculation must be performed before the cursor is displayed on the screen, is an example of key echoing requiring lesson material.

The largest single use on PLATO of these inter-active capabilities appears to be in game-playing programs such as "DOG FIGHT". There is little justification for designing the PDP-11 system to support game playing lessons. However, some lessons will need heavy student-computer interaction, and, therefore, an approach to supplying the required level of interaction has been selected which will not significantly increase the overall software complexity. A lesson requiring heavy interaction will be kept in central memory continuously while it is executing in the portion of the lesson requiring this level of interaction. In RSX-11D terms, the task executing the lesson will be "fixed" in memory while the student is in the highly interactive section of the lesson.

In order to treat lessons in this special manner without disrupting service to other users, the following restrictions are essential:

1. Only one task may be fixed in memory at a time. Since RSX-11D is a multiprogramming system, other lessons will continue to run, although with a longer response time.
2. The author must bound the "critical region" of the lesson requiring heavy interaction with TUTOR directives. The lesson will only be fixed in central memory while this portion is being executed. When a lesson is ready to enter a critical region and another lesson is fixed in memory, the lesson wanting service will not execute until it can be made resident in memory.
3. The critical region of the lesson must be fully contained within one lesson segment so no swapping will be required. Since a segment will contain at least a main unit and all the attached units this restriction should not have a major effect since heavy interactions are usually confined to a small part of the lesson.
4. A task cannot remain continuously fixed in memory for more than an installation-defined time period. At the end of this period, the lesson will be swapped out and will execute as if the TUTOR directives were not included in the program.

In spite of these restrictions we believe that the educationally meaningful lessons which truly require heavy interaction can be presented with a tolerable disruption to other simultaneous users.

6.5 Support of Execution of Non-TUTOR Programs

Some potential users have indicated a need to author TUTOR lessons or to execute computer-managed instruction material while TUTOR lessons are being presented. These additional functions can be supported under RSX-11D in one of two ways:

1. These programs execute as tasks with a lower priority than the programs required to support lesson presentation. Therefore, during busy periods these programs would run using whatever computer time was not required by other tasks.
2. One of the partitions is used only for running these programs.

The first approach has the disadvantage of being oriented toward using otherwise "wasted" computer time, but computer time is not expected to be a critical resource in this system. Swapping in a large program such as the TUTOR compiler will involve a lengthy disk transfer and the compiler is likely to use only a small amount of execution time before a higher priority task preempts its use of the partition.

The second approach which essentially provides a dedicated "background" partition will require a larger fraction of the available computer resources, especially central memory, but will reduce swapping activity. During periods of the day when only a few students are executing lessons this appraoch is probably feasible as long as a highly interactive lesson cannot be fixed in memory while these background jobs are being executed. The PDP-11 system is expected to have no more than three, and possibly only two, multiprogramming partitions. If an interactive lesson was fixed in one partition, while another partition was dedicated to uses other than presenting lessons, a large fraction of the computer resources would be serving a small minority of users.

The problems associated with the system providing different services simultaneously should not be allowed to significantly complicate the system software. The best way to resolve the simultaneous use issue is by operational procedures. We expect that the system will be heavily loaded with students during certain hours. However, at other times, the student load will be very light and the processing required for computer-managed instruction can be performed. Authors who wish to interactively compile and test new lessons will find the best system response for compilations when few other users are on the system, and authors may prefer to sign up for the machine when the only other user is some type of background processing, such as is required for computer-managed instruction. SofTech expects that most lessons presented on the PDP-11 system will be developed on larger systems such as PLATO, and the principal use of the smaller system will be for lesson presentation plus some computer-managed instruction. However, the PDP-11 based system will have the capabilities required to modify locally the lesson material distributed from other sites or to develop entirely new lesson material.

Section 7

SYSTEM DESIGN

The principal inputs to the system design activity were the strategies presented in Section 5 for handling TUTOR lesson material and data, the analysis presented in Section 6 of the expected disk usage, the PDP-11 hardware characteristics, and the properties of the RSX-11D operating system. Due to the limited scope of this study, the design was only carried to the level needed to insure that the system concept was technically feasible and to estimate the hardware configuration required to support lesson presentation to 24-32 terminals. The areas examined in the most detail in completing the technical feasibility evaluation were:

1. Central memory requirements necessitated by the need to keep the disk usage within the limits discussed in Section 6.3.
2. Approaches to implementing the required software within the RSX-11D task structure.
3. The enhancements to RSX-11D which would be required to insure the effective use of the PDP-11/45 for presenting TUTOR lessons.

7.1 Central Memory Requirements

The PDP-11 based system will be technically feasible only if a suitable substitute for the ECS electronic swapping memory is provided through:

1. Improved strategies (Section 5) for reducing the number of transfers from swapping memory.
2. An efficient implementation of the program and data swapping mechanism so that any frequently used information can be obtained from swapping memory or written to swapping memory with only one disk access and one disk transfer. Such a facility is provided by RSX-11D, as described in Section 7.2.

3. Adequate central memory to allow frequently used data for all active terminals to be kept resident at all times, thereby reducing the amount of information to be swapped when an interaction is to be processed.

Meaningful estimates of core memory requirements could not have been made without completing the software design if it were not for the detailed information on program sizes provided by the PLATO Systems Group. Although this data is the basis for all the estimates in this section, a significant margin for error must be assumed in making an evaluation of the system's technical feasibility since:

1. The PLATO data is in terms of 60-bit CYBER words and the expansion factor for estimating the PDP-11 values is uncertain. SofTech has used a factor of 5-6 for instructions and a factor of 4 for data based upon a small set of benchmarks.
2. The hardware architecture of the PDP-11 is significantly different than that of the CYBER and the methods used to structure the TUTOR interpreter may vary significantly between the two machines. Since the interpreter is the largest program and key portions of it will be central memory resident, size estimates derived from the PLATO data may be particularly subject to error.
3. The code for the PLATO System has been developed by a small group of excellent assembly language programmers over a period of several years. The code is "well polished" and an implementation on the PDP-11 of the TUTOR interpreter and command execution code is very likely to be larger than one might predict based on the machine comparison benchmarks, at least until a significant degree of "polish" has been applied to the PDP-11 system through performance tuning.
4. The amount of central memory required for temporary work areas is hard to estimate, especially since some of the PLATO programs may use ECS memory instead of central memory.

SofTech has estimated the PDP-11 central memory requirements by making maximum use of the available PLATO data and applying a 5-1 expansion factor for instruction and a 4-1 factor for data. However, the core requirements obtained in this manner for programs (except for DEC supplied software) have been further increased by one third since the estimating approach used is expected to underestimate core requirements.

In Exhibit 7.1 the core memory requirements are summarized. The RSX-11D system control tasks have no direct analogy with an element of the PLATO System; their size has been estimated based upon the functions they perform, as summarized in Exhibit 7.3.

The most questionable estimate is the size of the interpreter framework exclusive of the code to process each command. The framework will validate that the syntax of each command is correct, generate any error messages, insure that the use of the command is allowed by the state of the lesson, and then transfer control to the required command execution code. Some of these functions are not easy to isolate in the PLATO system, and differing estimates were made of what the size of the interpreter framework would be if all the code was gathered together.

The estimate assumes that two lessons must be able to be multiprogrammed at one time in order to support highly interactive lessons. When one of these lessons is fixed in central memory, other lessons must be able to be multiprogrammed. In addition, space is allocated for loading and executing infrequently used command code; each command being processed in this space, such as judging, using a vocabulary, graph plotting, output of a new character set to the terminal, etc. will run to completion to minimize swapping.

When TUTOR authors are developing new lessons or programs for computer-managed instruction are being executed, these functions will use the partition allocated to executing one lesson and possibly the partition used for handling the infrequently used commands. While only one

Exhibit 7.1

ESTIMATED CENTRAL MEMORY REQUIREMENTS
FOR 32 TERMINAL SYSTEM

Programs		16 Bit Words
RSX-11D Executive		24K
System-Control Tasks (Resident):		
Dispatcher		1K
Inter-lesson transition handler		1K
Terminal output		3K
Terminal input		3K
Student login		2K
Interpreter framework		10K
Resident command execution code (20 most frequently used commands)		8K
RSX-11D System Library (I/O, etc.)		4K
Resident Data		
Student data, terminal/lesson status 32 @ 1.1K/terminal		35K
Output buffers (estimated at average value of 128 words per terminal)		4K
Input buffers, 32 @ 75 words/terminal		2K
Micro tables (1 system, 1 user)		2K
Definitions of line drawn characters		2K
Data and programs per lesson multiprogrammed		
2 lessons @ 13K/lesson		26K
Estimated as:		
Lesson material	2K	
Command execution code	3K	
Work space for commands	2K	
Space for COMMON or extra storage data if segment requires it	6K	

Exhibit 7.1 (Continued)

ESTIMATED CENTRAL MEMORY REQUIREMENTS
FOR 32 TERMINAL SYSTEM

		16 Bit Words
Infrequently used command execution code		13K
Estimated for response analysis using a vocabulary as:		
Command execution code	6K	
Vocabulary data	5K	
Judging buffer and work space	2K	
	13K	
Other infrequently used commands will use less space.		
Total		140K
Allowance for unknowns in estimating program sizes (33%) (exclusive of RSX-11D executive system library, and resident data)		32K
Total Estimate		172K

lesson partition is being used for multiprogramming, interactive lessons will not be able to be fixed in central memory.

The estimate implies that the system is technically i[illegible]ible since only 124K of core memory can be attached to the PDP-11/45. However, there is a viable, although more costly, approach employing a dual processor system since a fairly sharp distinction exists between the programs and data required for lesson execution, and the programs and data required to handle input and output. The system configuration, described in more detail in Section 7.3, includes a PDP-11/45 with 124K of memory for lesson execution and a PDP-11/40 with 64K of memory for handling input and output. The data for inactive terminals is kept in the memory of the input/output processor until it is required for executing a lesson. Exhibit 7.2 shows the program and data included in Exhibit 7.1 which are allocated to the PDP 11/40. The functions of the input-output processor would include:

1. Input buffering and determining when a response is to be judged, i.e., when:
   a. Every key is to be judged
   b. The length limit of a response set by the LONG command is reached
   c. NEXT, or a key set with the JKEY command, is hit.
2. Key echoing in all cases except when lesson material is required. Micro tables would be handled in the 11/40.
3. Processing of the ERASE and EDIT keys.
4. Notification of the 11/45 when an interaction requiring lesson material is ready to be processed.
5. Output buffering and spooling of lengthy output sequences to the swapping disk.
6. Graphic output formatting, given a data structure specifying the graphic output. This function may require more central memory than is estimated in the 11/40, depending upon the amount of graphic processing done in the 11/45 (i.e., whether a circle is approximated by straight line segments in the 11/45). Some functions, such as this one, can be reasonably performed on either processor.

7. Generation and rotation of line drawn characters. This could be regarded as a special case of item 6.

With this distribution of functions, the central memory requirements of the 11/45 should be reduced to less than 124K. If a problem still exists, other storage saving measures could be employed; one example is reducing the total of COMMON data and extra storage data from 1500 to 1000 60-bit words, resulting in a saving of 1000 PDP-11 words for each of the two lessons being multiprogrammed. Therefore, the two processor system provides a viable technical solution to the memory problem while providing a natural division in system function that may reduce the overall software complexity.

Exhibit 7.2

ALLOCATION OF PROGRAMS TO THE PDP11/40
INPUT-OUTPUT PROCESSOR

Programs	
Executive	3K
Terminal Ouput	3K
Terminal Input	2K
All Resident Data	45K
Total	53K
Allowance for unknowns in estimating program size (33%)	3K
Total Estimate	56K

7.2 RSX-11D Task Structure

The central memory requirements were estimated without a detailed consideration of how each of the system functions would be implemented as RSX-11D tasks (Section 6.2). The most difficult problem in developing the task structure is that a task cannot address more than 32K of memory. This limit is a major problem in the case of the lesson presentation tasks, which must include:

	Size in PDP-11 Words	
Programs		
Interpreter framework	10K	25K
Resident command execution code	8K	
RSX-11D system library	4K	
Command execution code	3K	
Data		
Lesson material	2K	11K
Workspace for commands	2K	
Student variables/status variables	1K	
Space for COMMON/extra storage (if required) or extra space for lesson material	6K	
Total	36K	

These storage estimates do not include the factor previously included to compensate for estimating unknowns. Conceivably, the storage requirements could be reduced to 32K by clever programming, or operating parts of the interpreter as two communicating tasks. Such a "makeshift" approach is not adequate when the technical feasibility of the overall approach is being evaluated.

As noted in Section 6.1, the 11/45 hardware allows 64K of address space to be accessed at one time, 32K of instruction space and 32K of address space. However, the RSX-11D operating system does not currently take full advantage of the hardware facilities and restricts a task to a total of 32K of address space. Digital Equipment is likely to remove this restriction within RSX-11D in a future release of the system, although no formal commitment to this modification has yet been made. In SofTech's opinion the technical feasibility of a PDP-11 based system depends upon this modification to RSX-11D.

Each TUTOR lesson will be generated as an RSX-11D task consisting of a small root segment and several overlay segments. The root segment will consist of a small amount of code which will be executed when the task is invoked. The root segment will be invoked with the

identification of the terminal for which the lesson presentation task is being executed and will request a transfer of the terminal's student data and terminal/lesson status from the memory of the 11/40 (input-output processor) via the UNIBUS Link to the memory of the 11/45. This data will include the Lesson Description Table which will be used to load the appropriate lesson segment (Section 5.2) in order to continue the lesson at the point specified by the lesson status data. Each lesson segment will be an RSX-11D overlay segment and will be loaded by code in the root segment using the RSX-11D manual-load capability. COMMON and extra storage data if required will be read from swapping memory before execution of the lesson segment begins. The code for the interpreter framework and the resident interpreter commands will be installed in a shared library so only one copy is used by the two lesson execution tasks.

All status information and other data that will be required to continue the lesson will be part of the lesson/terminal status data to be stored in the 11/40. Therefore, RSX-11D will not be required to checkpoint lesson execution tasks and none of the programs will have to be swapped out; COMMON and extra storage data will have to be written to swapping memory if they were modified.

If the lesson uses one of the less frequently used commands which is to be loaded upon demand and which is executed in a separate partition (i.e., response analysis using a vocabulary), the task which executes the command is invoked via a request to RSX-11D from the lesson presentation task. The data required by the command is passed to it from the lesson presentation task via RSX-11D Global COMMON. The lesson presentation task is suspended until the task processing the command completes, thereby allowing the partition to be used to execute a lesson for another terminal.

Communication between the 11/45 and the 11/40 input/output processor is via the Terminal Output Task and the Terminal Input Task. When output to a terminal is required, the lesson presentation task will invoke the output task and will use the RSX-11D SEND mechanism to provide it with the terminal identification, the amount of output and the location in RSX-11D Global COMMON of the output. The output can then be transferred to a buffer in the 11/40 via a core-to-core transfer

controlled by the UNIBUS link. The Terminal Input Task will handle all terminal input from the 11/40. All input will be passed to tasks via the RSX-11D SEND mechanism if a small amount of data is involved or via RSX-11D Global COMMON if a large amount of data is involved.

The functions of the principal RSX-11D tasks are summarized in Exhibit 7.3. Additional tasks are likely to be identified during the detailed design of the system.

The JUMPOUT task, whose name is derived from the TUTOR JUMPOUT command, is particularly important since it handles all transitions from one lesson to another. When the student logs onto the system he is in the login lesson and a transition to the lesson to be studied is handled by the JUMPOUT task. The following steps are required before the appropriate lesson presentation task can be executed:

1. If the lesson is not on the swapping disk, transfer it from the disk used for the long-term storage to the swapping disk.
2. Move the TUTOR student variables to the 11/40 central memory if this is the first lesson being executed by the student.
3. Install the lesson as an RSX-11D task so that it can be invoked by the JUMPOUT task.
4. Transfer the data used by the lesson (i.e., COMMON data, extra storage data, vocabulary data, Charset data, micro table) to the swapping disk.
5. Initialize the Lesson Description Table to record the location of the lesson segments and data on the swapping disk and transfer it to the 11/40 central memory.

When these steps are completed, the JUMPOUT task will send a "startup" request to the lesson via the Dispatcher Task.

All lessons and data on the long-term storage disk will be managed by the RSX-11D file system. Data on the swapping disk will not be accessed using the file system since this data must be moved into central memory with one disk access and one disk transfer, and RSX-11D only supports multiple block data reads via low-level I/O routines. The Lesson Description Table will be used to access all data on the swapping

disk and the data will be loaded by routines which use the multiple block data read facility. The overlay segments of a task are loaded by RSX-11D using this facility and therefore no extra software development is required to handle TUTOR lesson segments.

Exhibit 7.3

PRINCIPAL RSX-11D TASKS

Task	Key Functions
Dispatcher	Assigns task priorities and time limits Initiates lesson presentation tasks
Terminal Input Control	Receives all input from I/O processor Invokes Dispatcher to schedule lesson presentation task
Termina. Output Control	Controls all output to I/O processor
Jumpout	Accomplishes all transitions from one lesson to another Moves lesson from long-term storage to swapping disk and installs it as a task Removes lessons not in use from swapping disk
Lesson Presentation	One such task exists for each TUTOR lesson and is invoked to execute the lesson by the Dispatcher task
Run-Time Arithmetic Compiler	Translates student-entered calculational expressions
Graphic Output	Generates all special display output such as: line-drawn characters circles axes, labels, etc.
Judge	Performs judging functions requiring vocabulary and/or concept data
Load Character Set	Loads the plasma panel terminal with a new character set
Student Data Logging	Records key data about a student session on long-term storage

In summary, RSX-11D provides an adequate base for developing the system if it is modified to allow a task to address 32K words of instruction space and 32K words of data space. The support for multi-programming of many task with differing priorities is more than adequate, and the file system will conveniently support the long-term storage of lesson and data. A more flexible method of communicating arbitrary amounts of data between tasks would be desirable, but the minimal capabilities provided by RSX-11D are an adequate base for developing the facilities needed to present TUTOR lessons.

7.3 Hardware Configuration

The central memory requirements for the system have been a major determinant of the hardware configuration. Therefore, the configuration shown in Exhibit 7.4 is likely to change significantly if Digital Equipment announces new members of the PDP-11 line which can support more than 124K words of central memory; such an announcement is expected in the first quarter of 1975.

The two processor system is coupled via the UNIBUS Link which allows memory-to-memory transfers between the two computers and via the RJS04 fixed head disk. The control part of the fixed head disk is on the 11/45 UNIBUS and the data-only part of the disk is on the 11/40 UNIBUS. Although this approach may appear cumbersome, the alternative approach to multiprocessor communication using the UNIBUS Window was rejected since it requires 11/45 memory address space. If more than 124K of address space should become available in a new member of the PDP-11 line, the two approaches to multiprocessor communication should be re-evaluated.

The plasma panel terminal interface has been designed by Magnavox, but apparently has not yet been manufactured. The design is weak in some areas and an alternative interface involving a small computer such as PDP11/05 should be investigated. The Magnavox interface attaches to the UNIBUS and each of up to 32 terminals has a UNIBUS address. The interface provides the terminal with a 20-bit PLATO-like output word, given two 16-bit PDP-11 words. The 10-bit input datum and the terminal identifier is assembled by the interface into one

16 bit PDP-11 word. Both input and output is interrupt driven and the 11/40 will have to service about 400 interrupts/second from 32 terminals in order to provide the interface with output data. This estimate is based upon PLATO statistics which indicate that an average 240 bits per second of output will need to be transmitted to each terminal.

The projected hardware cost for a 32 terminal configuration is summarized in Exhibit 7.5. Estimated costs are provided for the Magnavox interface, the Varian hard copy output device and the plasma panel terminals; the estimated cost of the terminals is intermediate between the price currently being paid by the University of Illinois for large quantity shipments and that being paid by the Air Force Advanced Instructional System Project for smaller quantities of terminals having special features.

The 64K words of central memory on the 11/40 includes 8-10K words in excess of the estimate in Exhibit 7.2. However, additional memory will be required during system development to support the testing program (Section 3), especially the simulation of input from one or more terminals (Section 3.4).

The configuration shown in Exhibit 7.4 is expected to support 24-32 terminals and additional terminals could be added only by increasing the amount of central memory, unless a higher performance device were to become available which could replace the fixed head disk. The use of a bulk core memory, or possibly some type of random access solid state memory, for swapping memory would increase the number of terminals supported while reducing the central memory requirements.

7.4 Approaches to Reduced Hardware Configurations

If fewer than 24-32 terminals are to be supported, a less powerful and less expensive hardware configuration should be required. If no more than 8-12 terminals were to be supported at one time, what configuration would be required? Exhibits 7.6 and 7.7 present a configuration designed to support this number of terminals. The most important reductions are:

Exhibit 7.4

HARDWARE CONFIGURATION PROPOSED TO SUPPORT 24-32 TERMINALS

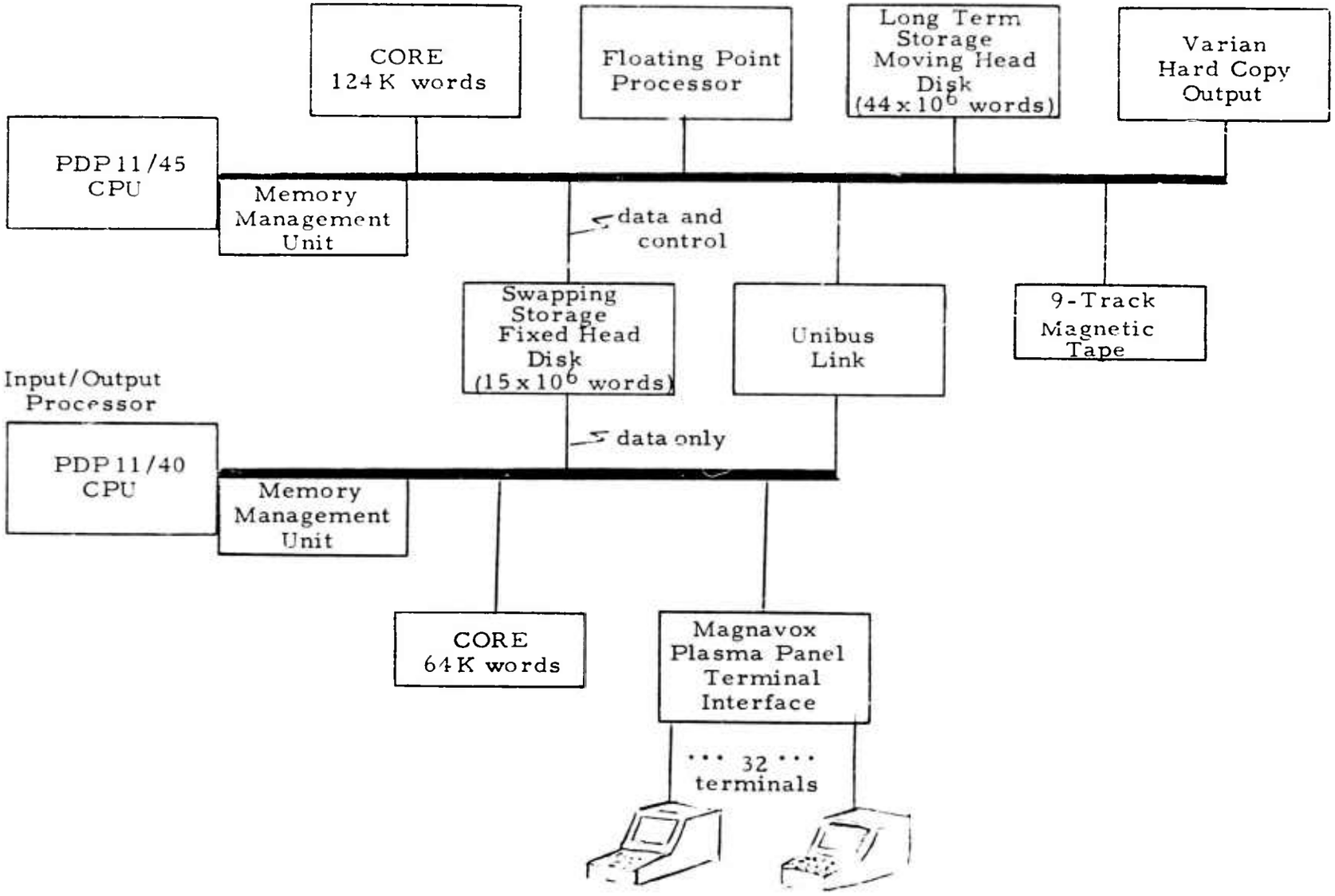


Exhibit 7.5

PROJECTED HARDWARE COSTS FOR CONFIGURATION TO SUPPORT 24-32 TERMINALS

CENTRAL SITE HARDWARE		
PDP 11/45 Central Processor with 32K words parity core memory, memory management unit	$37,570	
96K words parity core memory	$35,700	
Floating Point Processor	$ 5,290	
Fixed Head Disk (1.5 million word capacity)	$44,000	
Moving Head Disk (44 million word capacity)	$32,000	
Tape Unit (9 track, IBM compatible)	$10,745	
Cabinets	$ 2,400	
RSX-11D Operating System License	$ 5,000	
UNIBUS Link	$ 3,500	
PDP 11/40 with 16K core memory	$15,500	
Memory Management Unit	$ 2,480	
48K Core Memory	$14,300	
Varian Hard Copy Output Device	$12,000 (est.)	
Magnavox Terminal Controller	$20,000 (est.)	
Total Central Site Costs	$240,485	
TERMINALS		
32 Magnavox terminals @ $7,500 (estimated)	$240,000	
Total		$480,485

Exhibit 7.6

HARDWARE CONFIGURATION PROPOSED TO SUPPORT 8-12 TERMINALS

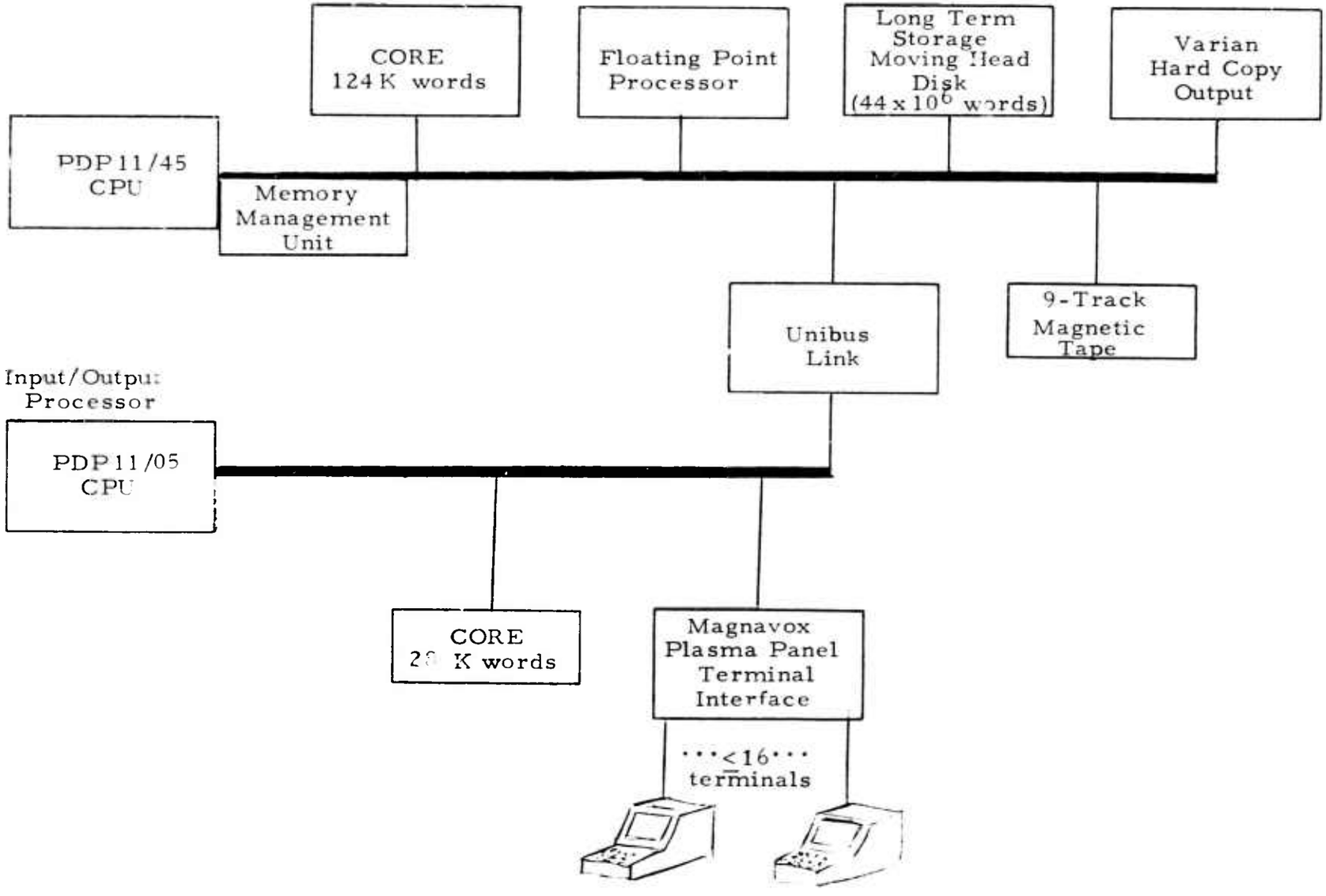


Exhibit 7.7

PROJECTED HARDWARE COSTS FOR
CONFIGURATION TO SUPPORT 8-12 TERMINALS

CENTRAL SITE HARDWARE		
PDP11/45 Central Processor with 32K words parity core memory, memory management unit	$37,570	
96K words parity core memory	$35,700	
Floating Point Processor	$ 5,290	
Moving Head Disk (44 million word capacity)	$32,000	
Tape Unit (9 track, IBM compatible)	$10,745	
Cabinets	$ 2,400	
RSX-11D Operating System License	$ 5,000	
UNIBUS Link	$ 3,500	
PDP11/05 with 16K core memory	$ 7,495	
16K Core Memory	$ 4,900	
Varian Hard Copy Output Device	$12,000 (est.)	
Magnavox Terminal Controller	$15,000 (est.)	
Total Central Site Costs	$171,600	
TERMINALS		
10 Magnavox terminals @ $7,500 (estimated)	$ 75,000	
Total		$246,600

1. The replacement of the PDP 11/40 by an 11/05. The amount of input/output processing is less and the ability to address more than 28K is no longer required.
2. The amount of core storage required on the input/output processor. The amount of resident data is proportional to the number of active terminals.
3. The fixed head disk is eliminated and the moving head disk which has the same transfer rate, but a longer access time (36 milliseconds, versus 9 milliseconds), is used for both swapping storage and long-term storage.
4. The number of terminals supported by the Magnavox interface.

The amount of central memory on the 11/45 has not been decreased because of the increased access time for the swapping memory and because some of the functions previously performed in the 11/40, such as graphic formatting, are likely to be shifted to the 11/45 due to the 28K limitation on the amounts of memory which can be attached to an 11/05. Reducing the 11/45 central memory to 92K may be feasible and this change would further reduce the cost by $15,000.

Further reductions in the system configuration are not likely to be cost effective since significant changes in the software design would be required if, for example, the input/output processor was eliminated or the 11/45 was replaced by an 11/40, thereby requiring software simulation of 64 bit floating point and 32 bit integer arithmetic. SofTech believes the concept of a PDP-11 based TUTOR system is likely to be cost effective only if at least 8-10 terminals are to be supported. The software development costs (Section 8.3) will be virtually impossible to justify if the system will only support two or three terminals.

Section 8
DEVELOPMENT PLAN

The development of the PDP-11 based system will required a substantial software development effort even if the RSX-11D Operating System is an adequate system base. In this section a development plan and deliverable item schedule is presented and the cost and the schedule implications of the plan are discussed. Although more detailed design work must be completed before accurate software development estimates are feasible, SofTech estimates that the projected software development work will require approximately 12-16 man years of effort and will require 24-30 months to complete.

The development plan includes a detailed design stage during which:

1. The semantics of the TUTOR language will be documented to the level of detail necessary for a reimplementation of the language.
2. The extensions to TUTOR recommended in Section 9 will be documented.
3. The software design will be documented in the "System Architecture" deliverable item at a level describing each RSX-11D task and the control flow and data flow through the two processor system. In particular, the function of each major subroutine in each RSX-11D tasks will be specified as well as all data which is shared between tasks.
4. RSX-11D will be used in order to test all system facilities important to the design.
5. A simulation model will be built of the combined hardware/software system in order to predict average response times (the time between the user hitting a key on the terminal and the output invoked appearing on the screen) with different assumptions about:
   a. The number of active terminals.
   b. The speed of the input/output processor.
   c. The amount of central memory available.
   d. The degree of multiprogramming supported among lesson presentation tasks.

6. A detailed development plan will be completed showing how the system will be implemented, validated, performance tested, installed, documented and maintained.

7. A detailed cost and schedule estimate will be prepared corresponding to the development plan.

The development of the simulation model, although not essential to the design and implementation effort, will focus attention on the assumptions made in this study and should provide a much more solid understanding of the key design issues.

At the completion of the detailed design phase, a final decision on whether to implement the system can be made based upon a more precise estimate of the software costs. The development of the TUTOR Language Reference Manual will have revealed any hidden complexities in the TUTOR languages. The RSX-11D testing program should have revealed any problems with the reliability of system as well as any problems with the facilities required to implement the design.

The detailed design phase is estimated to require approximately 6 months and approximately one and one half man-years of labor.

8.1 Detailed Design Phase - Deliverable Items

Exhibit 8.1 presents the key deliverable items to be produced during the detailed design phase. Work can proceed on the first four deliverable items in a highly parallel mode. The first three items must be completed before development plan can be completed and the detailed cost and schedule estimate prepared. The Implementation Phase is not scheduled to start until after the Detailed Design Phase is completed and the results evaluated by ARPA. The technical feasibility of the project and the implementation schedule assumes that Digital Equipment Corporation has modified RSX-11D to support separate instruction and data spaces by the end of the detailed design phase.

8.2 Implementation Approach

The implementation plan assumes that a high level system implementation language is used to develop a large fraction of the software in order to increase programming productivity and to reduce software maintenance and enhancement costs. The use of a high-level

Exhibit 8.1

DETAILED DESIGN PHASE

DELIVERABLE ITEMS

TUTOR Language Reference Manual

A detailed description of the semantics of the TUTOR language precisely defining the effect of executing each of the approximately 200 TUTOR commands in the author subset; estimated length: 400-600 pages.

Systems Architecture Technical Report

A technical presentation of the software design including the system control and data flow, the function of each RSX-11D task, intertask communication, and the principal processing steps in each task; estimated length: 75-100 pages.

Simulation Model Technical Report

A summary of the results obtained by use of the simulation model, as well as a description of the model characteristics and the sensitivity of the model to changes in the assumptions implicit in the model. The report will contain the program listings for the model; estimated length: 30 pages.

Development Plan

The plan will describe how the system will be implemented, validated, performance tested, documented and maintained. A detailed test plan will be included as well as a phased implementation plan. Estimated length: 25 pages.

Detailed Cost and Schedule Estimate

language to develop the TUTOR compiler and the TUTOR interpreter should facilitate the incorporation of the many small changes which are certain to be required during the testing program and initial use in order to make the new version of TUTOR correspond to the PLATO version.

Due to the need to treat central memory as a critical resource throughout the design, the code generated by the high-level language compiler must be highly optimized to minimize central memory requirements. Performance tuning will be an important component of the implementation phase and some replacement of high-level language routines by corresponding PDP-11 assembly language routines should be expected. Timing measurements made during system execution will provide the data necessary to isolate the routines which are being frequently executed and which require an unreasonable amount of computer time.

The only high-level language available from DEC for the PDP-11 is FORTRAN, which is not suitable for implementing the TUTOR compiler and interpreter. SofTech recommends the use of AED (12) an ALGOL-based systems implementation language developed and maintained by SofTech. AED includes many extensions to ALGOL including pointer (address) and bit manipulation operators and contains the flow of control primitives required for structured programming. AED compilers exist for IBM 360/370 CDC CYBER and UNIVAC 1100 series computers and cross compilers, which run on one of the above host computers and generate code for another computer, have been developed for over a dozen target computers, usually minicomputers.

The host machine for the PDP-11 cross compiler could be a computer on the ARPA network for which an AED compiler already exists. If the TUTOR compiler is written in AED, it could be developed entirely on the ARPA network machine until the interpreter was ready for use of the PDP-11. When the interpreter was completed, the output from the TUTOR compiler could be transferred from the ARPA network machine to the PDP-11 for testing. In this way, the TUTOR compiler, which must be available if most of the other components of the system are to be tested, can be developed using an existing AED compiler on an ARPA network machine while the AED PDP-11 cross compiler is being developed and/or prior to the availability of the PDP-11/45 hardware. When the development of the TUTOR compiler is complete on the ARPA network

machine, the AED cross compiler will be used to transfer the TUTOR compiler to the 11/45. The generated PDP-11 code will be loaded with the required PDP-11 AED run-time support in order to produce a TUTOR compiler running on the 11/45.

The TUTOR interpreter, which would be more difficult to test on an ARPA network machine, will probably be integration tested only on the 11/45, although some unit testing may be feasible on the ARPA network machine.

The software development tools on large computer systems, such as an IBM 370 or a UNIVAC 1108 are significantly superior to those on the PDP-11, especially in the area of file systems and utilities. With these tools, an efficient Program Production Library can be developed which will provide effective configuration control over the entire software system both during development and throughout the maintenance and enhancement phases. In order to reduce the logistics problems associated with using an ARPA network machine for program development, the 11/45 system used for software testing must be attached to the ARPA network so that programs can be moved easily between the large computer where they will be stored and compiled, and the 11/45 - 11/40 system where they will be tested. SofTech expects that it will be cost effective to store the compiler and interpreter source code throughout development on the ARPA network machine with executable code for the PDP-11 being produced by the PDP-11 AED cross compiler or by a PDP-11 cross assembler.

8.3 Software Development Costs and Schedules

One of the most important results of the detailed design phase of the project will be a cost estimate and implementation schedule derived from a more in-depth specification of software to be implemented, as well as much more precise description of the TUTOR language. However, an estimate of the labor required has been prepared, based upon the experience of the PLATO Systems Group in developing the TUTOR compiler and interpreter and SofTech's experience with large systems projects and in developing AED cross compilers. Exhibit 8.2

Exhibit 8.2

DETAILED DESIGN AND IMPLEMENTATION PHASES
LABOR ESTIMATES

DETAILED DESIGN PHASE	Man Months
TUTOR Language Reference Manual	5-6
Systems Architecture Technical Report	4-5
Experiment with RSX-11D	1-2
Simulation Model	3
Development Plan and Estimate	2
Project Management (@ 20 percent)	2
Total	17-20

IMPLEMENTATION PHASE	
Preparation for Implementation	
Establish Program Production Library on ARPA network host	2
Design Reference Manual Development covering:	
TUTOR Compiler (50 pages) TUTOR Interpreter (150 pages) PDP-11 control programs (50 pages)	12
Implementation of PDP-11 AED cross compiler	6
Implementation of PDP-11 AED run-time support and debug-ging tools	4
TUTOR Compiler Implementation	
Development on ARPA-network host	12
Transfer to PDP-11 and retesting	2
TUTOR interpreter Implementation	
Development of overall framework	12
Command execution code	24
Validation and Acceptance Testing	
Compiler	2
Interpreter	7

Exhibit 8.2 (Continued)

DETAILED DESIGN AND IMPLEMENTATION PHASES

LABOR ESTIMATES

	Man Months
PDP 11/45 - 11/40 Control Program Implementation	12
System Performance Tuning	6
Support of Field Installation	9
User Documentation	18
Maintenance Documentation	12
Management at 20%	28
Total	168

summarizes the labor estimates which include a 20 percent allowance for project management. No estimates are provided for computer time or travel expenses since the implementation phase logistics are not defined. During the detailed design phase these estimates will be refined and the assumptions on which they are based will be validated. In addition, alternative approaches will be developed if the study shows that implementation costs could be reduced significantly by not supporting in the PDP-11 system a given TUTOR feature, such as key echoing requiring lesson material.

The detailed design phase is estimated to require 1.5 man years, including approximately 4-6 man-weeks of consulting by members of the PLATO Systems Group, and would take place over a six month period. The implementation phase, which will produce a system that has been acceptance tested and ready for use, is estimated to require 10 to 14 man years, including 2-4 man-months of consulting by members of the PLATO Systems Group, and will take place over a 16-20 month period. Therefore, assuming a four month evaluation period between the two phases, the development of the PDP-11 based system will require 26-30 calendar months to complete. The total labor required is expected to be in the 12-16 man-year range or $480,000 to $640,000, based upon an average cost per man-year of $40,000.

Section 9

RECOMMENDATIONS

The principal conclusion of this study is that a PDP-11 based system for presenting TUTOR lessons appears technically feasible and will support 24-32 terminals on a hardware configuration costing approximately $200,000. No features of TUTOR have been sacrificed in designing the system although some restrictions have been necessary on the number of simultaneous users making heavy demands for system resources. All differences expected between using TUTOR on the PDP-11 and on the PLATO system are summarized below.

The PDP11/70 computer system, scheduled to be announced in the first quarter of 1975, should increase the number of terminals which can be supported with a given hardware investment or, alternatively, reduce the hardware cost required to support a given number of terminals. Anticipated features of the 11/70 include core memory in excess of the 124K word limit on the 11/45 and an increase in the transfer rate of the fixed head disk. Therefore, the 11/70 will greatly reduce the risk that the system design might be technically infeasible since central memory space and disk traffic were the critical resources identified during the design effort.

9.1 Differences Between the PLATO and PDP-11 Based Versions of TUTOR

The differences between the PLATO version of TUTOR and the implementation proposed for the PDP-11 are not major and, in SofTech's opinion, will not affect the utility of the system. Each of the differences have been described previously but are summarized here; they can be divided into four categories:

1) Data representation
2) Command semantics
3) Language extensions
4) Implementation restrictions

The approach selected for representing TUTOR data on the PDP-11 is presented in Section 4.3. These differences from the PLATO version of TUTOR are:

1. Eight characters (instead of ten) are stored per student variable.

2. The allowable range of floating point numbers is decreased, but the precision is increased.

3. A smaller range of integer values can be stored; an integer is stored as 32 bits rather than 48 bits.

The only difference in command semantics currently anticipated are:

1. Bit manipulation operations will be defined only for bit strings containing 32 or fewer bits, the size of an integer value.

2. The length of an input response is limited to 150 characters.

The following TUTOR language extensions are anticipated:

1. TUTOR directives will be defined for use in enclosing the sections of all lessons which perform key echoing using lesson material or which require real time interaction with animated output (Section 6.4).

2. System variables will be defined for the following values:
   a) Number of bits per TUTOR word (64 on the PDP 11)
   b) Number of bits per character (8 on the PDP-11)
   c) Number of characters per TUTOR word (8 on the PDP-11)

   in order to facilitate the development of programs which could be transferred from the PDP-11 system to the PLATO system with a minimum conversion effort.

3. The DEFINE command will be able to specify, optionally, the length of an array and of a character string.

The latter two additions will provide the information necessary for the compiler to generate a storage map which will help the author avoid errors in using the same student variable for different purposes.

The following implementation restrictions do not affect the semantics of the TUTOR language but appear required to insure that a very few terminals will not utilize an unreasonable fraction of the system resources.

1. Only one terminal will be able to perform at a given time highly interactive processing, i.e., key echoing requiring lesson material or interacting with animated output (Section 6.4).

2. The elapsed time spent in a highly interactive portion of a lesson must be less than an installation-defined limit. If the limit is exceeded, the lesson will execute with a priority equal to all other lessons being executed (Section 6.4).

3. No more than two micro tables (the standard system table and one user-specified table) may be used simultaneously for key echoing.

4. Authoring of new lessons will be possible only if the number of active terminals is a small fraction of the number of terminals the system could support. (Section 6.5).

5. The total size of a main unit, all of its attached units and the infrequently used command execution code required to execute the units must be less than the implementation defined lesson segment length (Section 5).

6. The highly interactive portion of a lesson must be contained within one lesson segment (Section 6.4).

In addition, limiting the length of COMMON data and/or extra storage to less than 1500 60-bit words has been considered; however, this restriction does not appear to be of significant importance since very few lessons use over 500 COMMON and/or extra storage variables.

The net effect of these changes and restrictions on the typical TUTOR author should be small and few lessons now in use on the PLATO system should require program logic modifications if they are to be transferred to the PDP-11 based system. Conversion aids could be used to focus attention on the parts of a program likely to require change, such as units using the bit manipulation operators and units performing key echoing requiring lesson material.

9.2 Recommended Development Approach

SofTech believes the feasibility of a PDP-11 based system is sufficiently well-established and the above limitations sufficiently few to justify undertaking the detailed software design phase, which was described in Section 8, if

a) a requirement exists for presenting computer-aided instruction lessons at remote and/or classified sites,

b) more than one system is to be acquired, thereby spreading the cost of the initial software development effort over multiple systems.

A significant fraction of the software development effort is independent of the language used to develop lesson material. The requirement for presenting computer-aided instructional material at remote and/or classified sites may not be confined to lessons written in TUTOR, but may extend to other languages, such as CAMIL which is being used on the Air Force Advanced Instructional System. If so, SofTech recommends a study, as discussed in the following section, to examine how lessons developed in these languages could also be presented on the PDP-11 based system.

SofTech's pre-announcement information on the PDP-11/70 indicates that this machine will provide an excellent hardware base for the system and will avoid some of the problems associated with the 11/45 - 11/40 configuration, while providing substantially increased system capabilities at a slightly reduced hardware cost. An appendix to this report will be provided by SofTech following the announcement of the 11/40 which will summarize the impact of the 11/70 on the system design, the proposed hardware configuration and the hardware cost.

9.3 Areas for Future Research

The ability to present lesson material on the PDP-11 based system which is not written in TUTOR would greatly increase the utility of the system. The TUTOR language is frequently criticized because an author requires extensive programming experience in order to use it effectively. Simple TUTOR lessons can be written with very little experience, but the power of the PLATO system can be exploited only by those versed in its rich command repertoire. A survey of a group of popular lessons, which are generally regarded as good examples of computer-aided instruction, indicates that typical lessons use 47-79 of the approximately 200 TUTOR commands available to authors. Learning how to use that number of TUTOR commands requires more time than most authors are willing to invest.

Two different approaches are under development with the goal of reducing the effort an author must make to utilize a computer-based educational system. The CAMIL language being used on the Air Force AIS project is designed to support both computer-aided instruction and computer-managed instruction via author-oriented extensions to an ALGOL-like programming language. The extensions will be designed to provide English-like commands

for many frequently used operations. However, if a command is not available for the desired operation, a programming language of roughly the complexity of PL/I must be learned. Currently CAMIL does not have as rich a repertoire of commands as TUTOR and therefore many operations which are easy to perform in TUTOR are more difficult to specify in CAMIL. However, the CAMIL facilities for data base manipulation have no direct analogy in TUTOR, and are required for computer-managed instruction.

The TICS System (14) developed at the Massachusetts Institute of Technology with support from the National Science Foundation provides an interactive computer system to assist the author in developing lessons which involve a dialog between a student and the computer. Although the computer system facilities which can be used in presenting the dialog are more limited than those available via TUTOR, the author using TICS need not become a programmer. An interactive dialog will solicit from the author the required information to construct the control structure of the lesson and to specify the response judging appropriate at each point in the dialog. The TICS system generates all the information required for a computer system to present the lesson; the presentation need not be on the same computer system as the one on which the lesson was developed. TICS currently operates on the Multics time-sharing system at MIT, which is part of the ARPA network; a presentation system based upon a PDP-11 has been designed but not implemented. In SofTech's opinion the TICS concept of providing computer assistance in the authoring process, and of not forcing an expert in a subject area to become a computer programmer, can significantly expand the use of computer-based educational systems.

SofTech recommends that the work performed on this study be expanded in order to investigate the use of the PDP-11 based system for the presentation of lessons written in CAMIL and developed using TICS. The ability to execute CAMIL programs would provide the PDP-11 based system with a computer-managed instruction capability, while the TICS approach would allow low-cost generation of dialog-oriented lessons which do not need the highly interactive features of TUTOR. With multiple approaches to generating both computer-aided instructional material and computer-managed instructional material, SofTech believes the PDP-11 based system would prove valuable to a wide variety of agencies concerned with technical training, especially when the training must take place at remote or classified sites.

BIBLIOGRAPHY

1. B. A. Sherwood, The TUTOR Language, Computer-Based Education Research Laboratory (CERL), University of Illinois, 1974.

2. Bitzer and Johnson, "PLATO: A Computer-Based System Used in the Engineering of Education", Proceedings of the IEEE, Vol. 59, No. 6, (June 1971), p. 960-968.

3. E. Avner, PLATO User's Memo: Summary of TUTOR Commands and System Variables, CERL, University of Illinois, 1974.

4. E. Lyman, PLATO IV Curriculum Materials, CERL Report X-41, University of Illinois, 1974.

5. "PLATO IV Communications System", CERL, University of Illinois, 1975 (Draft report; for more information on availability contact PLATO Software Group, CERL, University of Illinois, Urbana, Ill. 61801).

6. P. Tenczar, "TUTOR Graphics Capabilities", Proceedings 1974 Society for Information Display International Symposium, p. 70-71.

7. Tenczar and Golden, Spelling, Word and Concept Recognition, CERL Report X-35, University of Illinois, 1972.

8. PDP 11/05/10/35/40 Processor Handbook, Digital Equipment Corporation, 1973.

9. PDP 11/45 Processor Handbook, Digital Equipment Corporation, 1974.

10. Introduction RSX-11D, DEC-11-OXINA-A-D, Digital Equipment Corporation, 1974.

11. RSX-11D Concepts and Capabilities, DEC-11-OXDCA-B-D, Digital Equipment Corporation, 1974.

12. Introduction to AED Programming, SofTech, 1973.

13. R. Kaplow et al, TICS: The Author Language and Instruction Manual, Massachusetts Institute of Technology, 1972; "TICS, A System for the Authoring and Delivery of Interactive Instructional Programs", Proceedings of the Seventh Annual Princeton Conference (1973).

Appendix A

PLATO-TUTOR DATA

During the study the data gathered by members of the PLATO Systems Group on various aspects of the PLATO system provided information essential to the design of the PDP-11 based system. Much of this information is not available in the literature and therefore has been summarized here with the hope that it may prove useful to the designers of future computer-aided instructional systems.

This appendix is divided into the following sections:

1. Overall System Storage Requirements
2. Data Requirements for Executing TUTOR Lessons
3. CPU Utilization and Command Usage Statistics
4. Properties of TUTOR Lessons
5. Response Time Characteristics

The use of this data requires a good working knowledge of the PLATO system and no attempt has been made in this appendix to interpret the data or to present it in a tutorial form.

A.1 Overall System Storage Requirements

The space requirements in Central Memory (CM) and in Extended Core Storage (ECS) are:

	Space in 60-bit Words	
Mercury Operating System	3K	CM
Overlays, Tables	100K	ECS
PLATO-TUTOR Operating System	15K	CM
Tables used by Operating System	10K	CM,
	100K	ECS
TUTOR Command Execution Code	10K	CM
	70K	ECS
Lesson Buffer (for lessons in use by some terminal)	600K	ECS
Data for 300 terminals (station/student banks)	200K	ECS

The key elements of the PLATO-TUTOR central memory tables are:

Student bank and variables (contains 150 TUTOR variables)	400
Judging buffers (allows up to 300 characters in a response)	800
COMMON buffer (holds COMMON and/or extra storage data)	1500
SHOW Command buffers	100
Output buffer (stores intermediate form of output)	300
ECS overlay buffer (holds dynamically loaded code)	1000
Work buffer (provides storage which is not preserved between time slices since commands, except for CALC, are not interruptable)	300
Storage for CALC execution (allows a large DEFINE set to be available at run-time)	2500

A.2 Data Requirements for Executing TUTOR Lessons

I. Data Per Terminal in Use (i.e., per student)

Station data - 32 words (includes keyboard input buffer)

Output Buffer - 16 words average per terminal

Status data (400 words)
- 150 student variables
- 250 station variables

A significant fraction of the status variables could be packed, but are unpacked for ease of access. Apparently approximately 125 words could be saved by packing.

Extra storage data (0 - 1000 words)

Router variables (0 to approximately 75)

II. Data Per Lesson

A lesson consists of units and may be shared between students. Average lesson is 50-70 units long; average unit is 50-60 words. Per interaction 3-4 different units are used, some several times.

COMMON data 0 - 1500 words CM; 0 - 8000 words ECS Under 25% of users require COMMON and majority of these require less than 1000 words. Sum of extra storage and COMMON data in central memory must be less than 1500 words.

Charset data - approximately 460 words per character set definition

Micro table - maximum 256 words per micro table

Vocabulary - 2,000 - 3,000 words

III. Data Per Course

This data is "owned" by the Router lesson and should be available only when the router lesson is active

Router lessons - size like other lessons

Named COMMON can be used in several lessons of a course

Leslist data - 644 words (two disks blocks) describing which lessons make up a course

A.3 CPU Utilization and Command Usage Statistics

I. Gross Statistics

An average of 16,000 CDC CYBER 73 instructions are used to process an interaction requiring lesson material; key echoing requires about 500 instructions.

Keys are input at the following rate:
1 every 4 seconds by students
1 every 2 seconds by authors
Every other key requires lesson material to process.

About 70% of one of the two CPU's is used for condensing lesson material.

II. Execution Time Distribution by Command

Only ten commands use over one percent of the total lesson execution time

Command	Execution %
At	4%
Calc	33%
Draw	14%
Goto	21%
Join	1%
Jump	1%
Showa	2%
Unit	3%
Write	2%
Writec	1%
Total	82%

All expressions evaluated via the CALC command are pre-compiled to CDC machine language.

Response judging requires only about 0.5%

The main interpreter loop and function key handler uses about 12%

Key echoing requires about 2.5%

A.4 Properties of TUTOR Lessons

I. Lesson Statistics

The following table shows the sizes of a representative sample of TUTOR lessons in terms of the number of lines of TUTOR source code, the number of TUTOR commands used and the amount of command execution code invoked to process the lesson.

Lesson	Field	Commands	Lines[1]	Size in 60-bit words[2]
SHOW	Physics (1st yr)	79	2058	7033
QUAL	Chemistry (1st yr)	54	1729	5165
FLY*	Biology	32	1818	1524
NMR	Chemistry(2nd yr)	30	206	4698
OPTA	Chemistry(2nd yr)	47	1880	5201
RALPH1	Russian (1st yr)	54	1421	3401
CELL*	Biology	38	1154	1740
TITRATE	Chemistry	67	1349	5664
MOTHS**	Biology	72	1787	6705
BIRDS	Biology	54	733	5134
WEST	Elem. Sch.	53	1222	5003
BAGGER	Elem. Sch.	63	2029	4876
VECTORS	Physics	47	1353	3782
POPGEN	Demography	55	1327	5288
POPULS	Demography	65	1580	5953

[1] of TUTOR Source

[2] of Command Execution Code

* old lesson - originally written for PLATO III

** old lesson - but updated for PLATO IV

II. Analysis of One Lesson

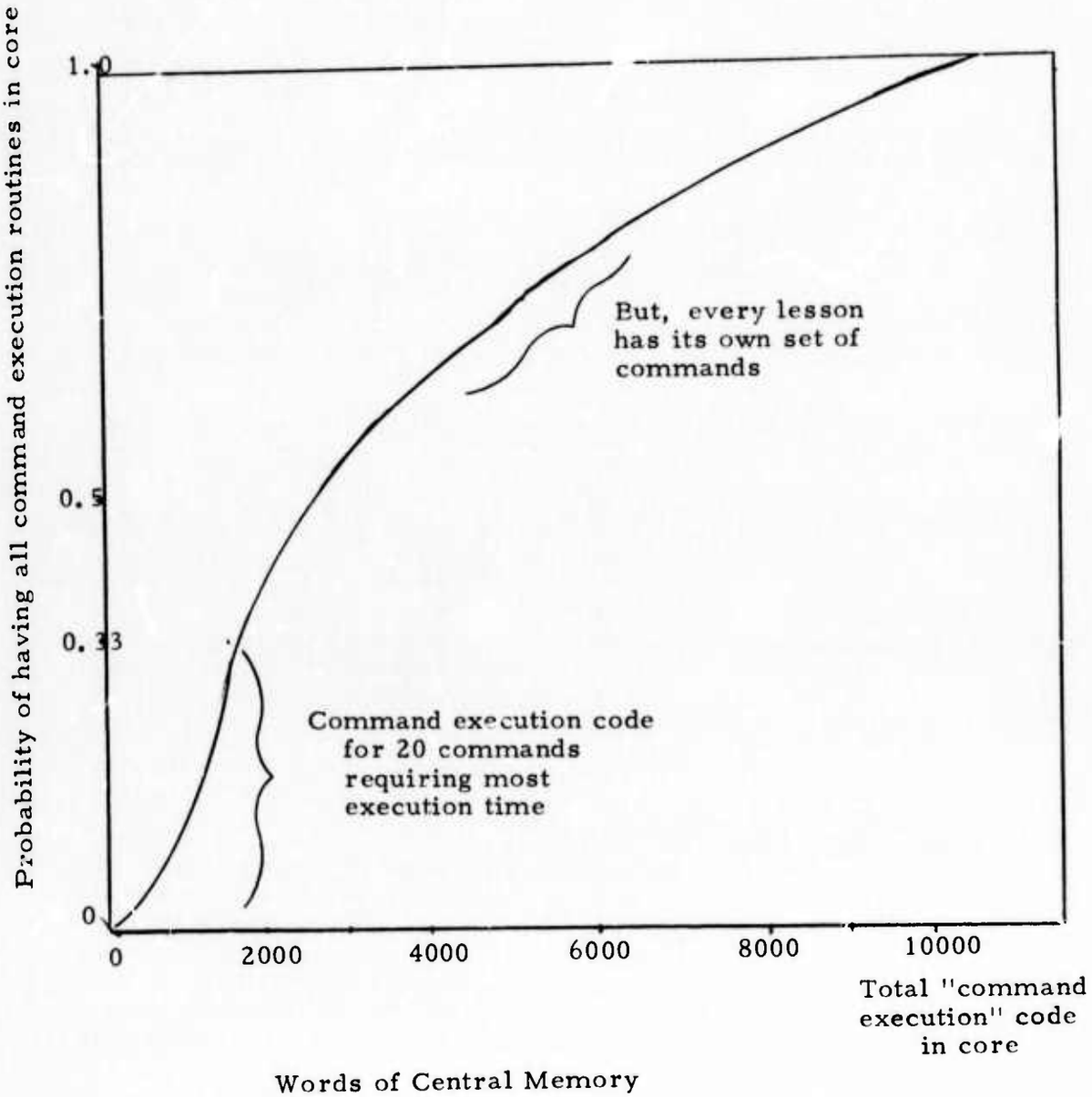
Lesson SHOW has 79 commands. It is a "good" physics lesson that teaches vectors. It uses about 1/3 of the total number of TUTOR commands. Most "good" recent lessons at the college level (e.g., physics, chemistry, biology) use about the same number of commands. The following table shows the number of times each command was used in the source code; the commands marked with an asterisk are the ten commands using 82 percent of CPU time on a system-wide basis.

command	# of references	command	# of references
add1	9	bump	2
ansv	38	*calc	117
answer	2	calcs	10
area	2	charset	1
arrow	47	circle	5
*at	332	compute	2
axes	2	data	1
back	7	data1	1
base	7	dataon	1
bounds	4	define	28

command	# of references	command	# of references
delay	1	origin	8
do	114	pause	1
*draw	30	polar	3
end	1	put	1
endarrow	38	putd	2
entry	3	randp	2
erase	43	randu	2
exact	3	restart	1
funct	1	rdraw	4
gdraw	5	rotate	7
* goto	26	scalex	5
graph	22	scaley	4
imain	1	search	2
inhibit	8	setperm	2
jkey	3	* showa	2
* join	93	show	3
judge	32	showt	3
* jump	9	size	9
labelx	2	specs	1
labely	1	store	3
locate	26	storea	5
long	1	storen	2
match	1	term	3
mode	45	* unit	81
move	1	vector	33
next	14	* write	527
nextnow	5	* writec	46
no	32	wrongv	14
ok	4	zero	2
or	1		

Approximately 40 percent of the commands are referenced only one or two times in the source code, indicating they appear in at most two units.

Of the 7033 words of command execution code used by the lesson, 1522 words of it correspond to the ten commands using over 80 percent of a typical lesson's execution time plus the next ten most frequently used commands. Each lesson requires a different set of the command execution routines, depending upon which of the infrequently used commands it requires as illustrated by the following graph.


Probability of having all command execution routines in core
1.0
0.5
0.33
0
But, every lesson has its own set of commands
Command execution code for 20 commands requiring most execution time
0
2000
4000
6000
8000
10000
Total "command execution" code in core
Words of Central Memory

The following cross reference tables shows that only a small number of the units are referenced frequently from other units, although the lesson is quite interactive and definitely not a "page-turning" lesson.

Physics

LESSON SHOW AT 11:06 AM ON TUESDAY, SEPTEMBER 17, 1974 PAGE

UNIT	BLOCK	UNIT LOCATION	REFERENCES TO UNIT
ANGLE1	ANGLE	1281	1335
ANGLE1A	ANGLE	1301	1335
ANGLE1B	ANGLE	1308	1309
ANGLE2	ANGLE2	1411	1412
ANGLE3	ANGLE3	1514	1515
ANGLE4	ANGLE3	1542	1543
ANGLE5	ANGLE3	1603	1604 1674
ANGLE5A	ANGLE3	1613	1604 1639 1647 1673
ANGLE5C	ANGLE5---	1619	1628
ANGLE5D	ANGLE5---	1649	1620
ANGLE5E	ANGLE5---	1671	1639 1672
ANGLE6	ANGLE5---	1689	
ANGLE7	ANGLE7	1728	
BADKEY	LERNMOVE	122	110
BLANK	MOVER	1972	143 271 294 345 424 448 471
			522 539 556 579 596 610 613
			1033 1059 1103 1126 1201 1216 1253
			1318 1455 1469 1493 1542 1706 1713
			1750 1763 1999 2039
CHAR	SHOWREG	1830	49 93 150 243 824 843 747
			771 807 826 860 1268 1466
CHKRULER	LERNMOVE	167	
CURSOR	MOVER	1942	1940
DATIL	MOVER	2053	
DIVIDER	LERNMOVE	134	96 647 773 804 828 1464 1555
DTOR	ANGLE	1345	1320
END	LERNMOVE	114	111
EVALIT	SHOWREG	1801	
GETRY	ANGLE2	1484	1432
GETVEC	MOVER	1992	686 691 698 749 849 1384
GETVECSP	MOVER	1986	1862
GETVEC1	MOVER	1998	1990 1996
GETVEC2	MOVER	2020	2007 2027
GETVEC3	MOVER	2032	2014 2020 2023 2030
GETVEC4	MOVER	2035	1865
IHOY	LERNMOVE	142	1954
IGIRL	LERNMOVE	137	97 485
IJNOTE	IJNOTE	745	237
INDEX	INDEX/VEC2	194	144 2054
INTRO2	INTRO	66	48 148 198
LAST	USETRIG	1919	
LERNMOVE	LERNMOVE	91	236 241
MESSAGE	MOVER	2049	1863
MOVER	MOVER	1931	104 257 410 493
NEGLONG	MOVER	2038	711 726 739
NOAUSE	MOVER	1977	74 330 336 371 380 383 394
			440 459 477 531 548 563 571
			584 599 617 632 653 656 664
			684 718 731 754 781 785 778
			793 813 818 838 862 866 872
			900 904 1043 1068 1092 1110 1130
			1155 1165 1169 1180 1185 1195 1213
			1286 1290 1297 1332 1418 1421 1437
			1443 1447 1461 1507 1527 1536 1543
			1583 1585 1700 1739 1742 1841 1873
			1880 1900 1905 1911

UNIT	BLOCK	UNIT LOCATION	REFERENCES TO UNIT
PICK	SHOWREQ	1805	907 909 958 960 1146 1148 1231 1233 1240 1338 1342 1422 1424
PICKD	ANGLE	1337	1311
PICKR	ANGLE	1341	1311
POINT	ANGLE3	1400	1582
PREANG	PYTHAG6	1267	237
PREPYTHA	SURV-TRIG	881	
PRODEG	ANGLE	1363	1282 1419 1289 1420
PRORAD	ANGLE	1384	
PYTHAG	PYTHAG	887	
PYTHAG2	PYTHAG	930	
PYTHAG2A	PYTHAG	944	1017
PYTHAG3	PYTHAG	955	931 993 1002 1006
PYTHAG3A	PYTHAG3A	1004	956 993
PYTHAG3C	PYTHAG3A	970	964 968
PYTHAG4	PYTHAG3A	1019	1005
PYTHAG5	PYTHAG4---	1144	
PYTHAG6	PYTHAG6	1227	
RTOD	ANGLE	1353	1320
RVECT	SHOWREQ	1772	1511 1779
RVECTR	SHOWREQ	1778	1491
RVECTR2	SHOWREQ	1782	1780
SCALE	LERNMOVE	128	95 173 646 772 803 827 845 1957
SETUP	INTRO	43	
SHOWREQ	SHOWREQ	1789	953 1307 1611
SHOWSCOR	SHOWREQ	1792	992 1001 1328 1638 1646
SURVEC	SURV-TRIG	824	237
TELERR	SHOWREQ	1810	920 996 1206 1223 1322 1500 1644
TRIG	SURV-TRIG	859	237
USETRIG	USETRIG	1843	237 1729
VECFQ	IJNOTE	770	237 801
VECINTRO	INTRO	47	45 68 236
VECNEG	IJNOTE	800	237 825
VECSUM	VECSUMMARY	623	236
VEC1	LERNMOVE	147	67 170 192 236
VEC2	INDEX/VEC2	239	119 236
VEC2B	MOVER	1954	244 391
VEC3	VEC2/VEC3	389	242
VEC4	VEC4	514	390
VEC4A	MOVER	1964	333 392
VEXAMPLE	VECSUMMARY	642	236
WTADUP	USETRIG	1893	
XYAXES	MOVER	1980	374 393 648 774 805 829 847 1874
ZAVEC	MOVER	1969	412 1965

A.5 Response Time Characteristics

The response time of the PLATO system is usually described as the total time for a key to leave the terminal, to be processed by the system and to have a response back at the terminal. A typical response time is in the 100-150 millisecond range; delays longer than about 200 milliseconds will be observable by a touch typist. Due to the design of the PLATO system the response time is not significantly affected by whether or not lesson material is required to perform key echoing since the student bank data must be retrieved from ECS memory to process every keypress. However in the PDP-11 based system two response times will exist:

a. a shorter time if the input/output processor can echo the key (most keys)

b. a longer time if the 11/45 must use lesson material to echo each key (a small fraction of keys)

The data observed on the PLATO system is:

Total time for a keypress to leave keyset, be processed by computer in Urbana, and have response back to terminal. (Don't touch anything...runs automatically via echo keys)

N = 72 Ave. Millisecs = 120 S.D. = 25

121	92	111
93	155	92
103	101	103
101	108	165
161	112	105
147	103	163
105	118	126
96	111	153
170	148	137
107	102	103
101	109	104
104	134	105
148	96	111
158	116	112
187	103	114
133	101	108
102	176	111
140	101	102
102	122	107
156	103	151
150	156	161
104	147	112
95	102	104
91	122	98

Appendix B

BENCHMARK COMPARISONS OF CYBER 73 AND PDP 11/45

Three benchmark algorithms were used to estimate the relative speed and program expansion ratios likely to occur when programs now existing on the CYBER 73 are re-implemented on the PDP11/45. The program expansion ratio is the number of PDP-11 words corresponding to one CYBER word of executable code. A 4:1 expansion ratio for data words has been used throughout the study based upon an evaluation of how data is represented on each machine.

The two benchmark algorithms used to estimate the program expansion ratio have been previously used by SofTech to benchmark avionics computer systems, an environment in which core memory requirements are particularly critical. The first benchmark, Unpack, places heavy emphasis on partial word and bit manipulation operations; similar operations would be heavily used in graphic output formatting and in many character manipulation routines. The second benchmark, Dispatch, emphasizes indexing through data.

A.1 UNPACK Benchmark

CYBER Description

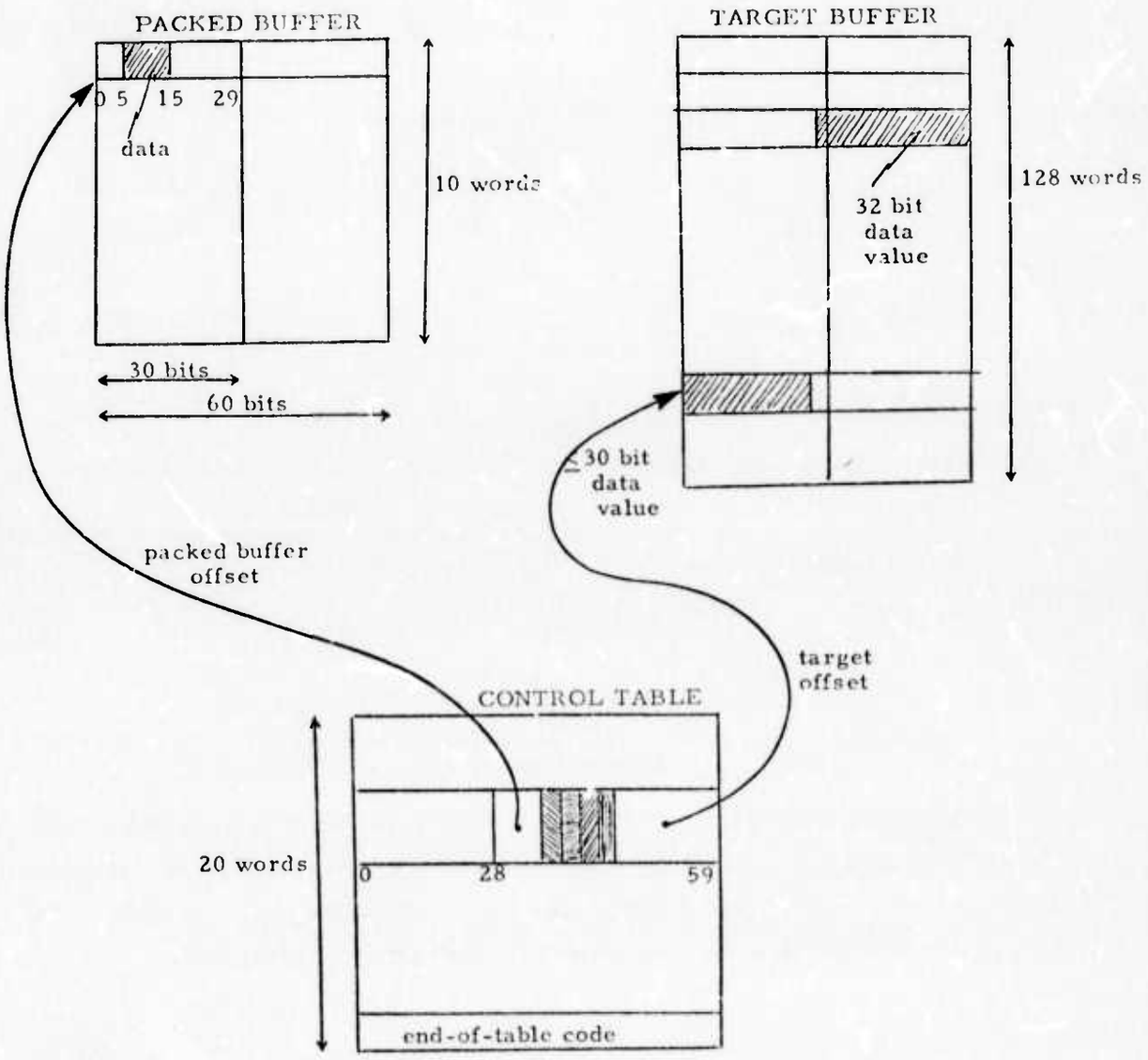
A buffer area of ten 60-bit words containing 20 values is to be unpacked into a target buffer of half or full words using a control table to describe the data to be unpacked. In the packed buffer data is stored in bits 5-15 of each halfword; no assumptions can be made about the unused bits. The packed data is to be placed in data fields of 1 to 32 bits right-justified in half or full words scattered through the target buffer of 128 words. The control table, which is composed of full words containing 32 bits of right-justified information has six fields which describe the data to be unpacked and how it is to be stored in the target buffer (See Exhibit B.1). The three tables are shown graphically in Exhibit B.2.

Exhibit B. 1

Bit Position	Function
27-35	Packed buffer offset (0-19) specifying half word to be unpacked.
36-43	Number of bits in the field in the target buffer (1-32). If more than 30, a full word is required.
44-47	Starting bit in the halfword being unpacked (5-15).
48-49	Sign to be applied to the data value in the target buffer (00 = unsigned, 01 = signed).
50-51	Describes whether a halfword or a full word is to be used in the target buffer. (00 = store in half word, 01 = store in full word) Must be a full word if bits 36-43 specify more than 30 bits are to be stored.
52-59	Target buffer offset in half words (0-255). For full word storage (bits 50-51 = 01), this location must be even.

The last word in the control table is zero, signifying the end of processing.

Exhibit B. 2



SOFTECH

PDP-11 Description

The packed buffer consists of twenty 16-bit words, in which the 11 data bits are right-justified. The target buffer consists of 256 16-bit words; a value requiring more than 16 bits will be stored in two words, starting at an even address. The control table consists of one entry for each word in the packed buffer; each entry consists of two 16-bit words:

		Location		Corresponding CYBER Field
Word 1	(bits)	0-7	(bits)	27-35
		8-15		36-43
Word 2		0-3		44-47
		4-5		48-49
		6-7		50-51
		8-15		52-59

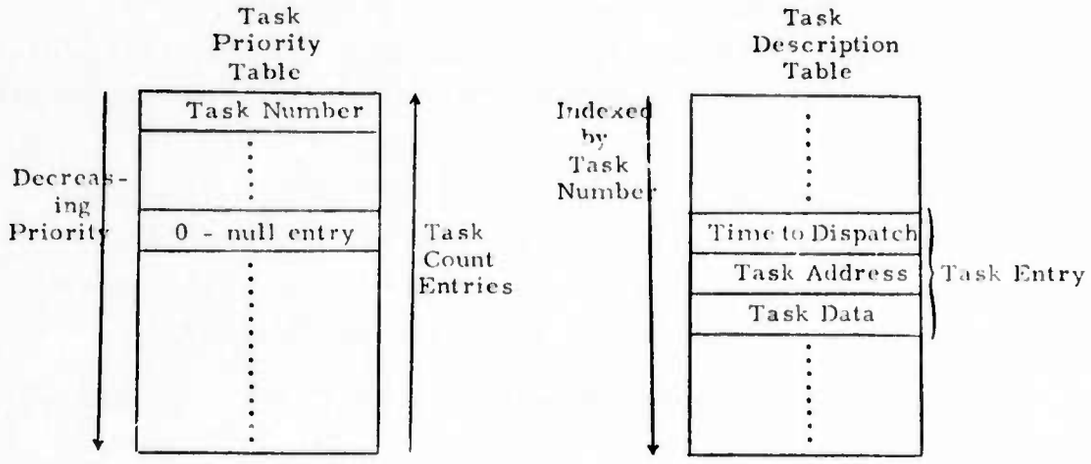
A.2 DISPATCH Benchmark

The benchmark performs the following repetitive actions, which are typically part of the dispatching algorithm, in an executive control program:

1. Read clock and update system time.
2. For each task,
   a. determine if the task is active
   b. if active, determine if it is time to dispatch the task and, if so, call the task.

The following data structures (See Exhibit B.3) are used by the dispatching algorithm. A task number of 0 indicates no task is assigned this priority, i.e., a null entry. The task data is not to be examined by the dispatcher but is to be passed to the task when it is called.

Exhibit B. 3



Dispatching is to be performed by scanning the entire list of tasks in priority order and comparing the system time read before beginning the scan ("now") with the time at which the task should next be dispatched, which is stored in the Task Description Table. If the task time is less than or equal to "now", the task time is set negative and the task is dispatched; otherwise, the scan continues. The task time will be reset by a scheduler, which is not part of the benchmark. When the executive regains control, the dispatcher always begins the scan with the highest priority task.

CYBER Description

The Task Priority Table entries are full words (60 bits). The Task Description Table stores the time as a full word, the task address and task data as two half words (30 bits).

PDP-11 Description

The Task Priority Table entries are 16-bit words. The Task Description Table stores the time as two words (32 bits), the task address as a 16-bit word, and the task data as a 16-bit word.

A.3 Benchmark Methodology

The CYBER code was written by an experienced CDC 6000 series assembly language programmer and is believed to be "polished" code that is comparable in quality to that existing in the PLATO system. The PDP-11 code was produced by hand compiling a program written in the AED language (Section 8.2) with the assumption that the same type of code optimization techniques would be used in the PDP-11 compiler as have been used by SofTech in implementing optimizing AED and JOVIAL compilers. The AED programs were hand compiled by the developer of several AED cross-compilers who predicted the type of code which it would be cost effective to generate with an optimizing PDP-11 AED cross-compiler.

The CYBER programs were written using special purpose calling sequences:

UNPACK	B1: Address of Packed Data Buffer B2: Address of Target Buffer B3: Address of Control Table B1, B2, and B3 are not changed within UNPACK. The index registers X1 and X2 are saved and restored. No registers are saved on entry or restored on exit.
DISPATCH	B1: Address of Task Priority Table B2: Address of Task Description Table B1 and B2 are not changed within DISPATCH. The task is called, without saving any registers, with the system time and the task data in an argument list whose location is specified by an index register.

The AED program for UNPACK used a standard calling sequence suitable for calling external procedures. Three arguments, the Packed Data Buffer, the Target Buffer, and the Control Table, were passed as static integer arrays. An AED calling sequence passes the address of an array.

The DISPATCH benchmark is assumed to be a block of in-line code. The task is called assuming it is an external procedure, with the system time and the task data as arguments.

A.4 Code Expansion Ratio Results

The following results for the code expansion ratio were obtained for the two benchmark algorithms:

	CYBER 73*	PDP11/45**	Ratio
UNPACK	41	222	5.4
DISPATCH	13	50	3.9

*60-bit words; **16-bit words

The two algorithms are obviously a very small sample on which to make any code size estimates for the PDP-11 based system, but a 5:1 code expansion ratio was selected for use throughout the study in scaling up PLATO central memory data. The central memory estimates described in Section 7.1 were further increased by one third due to the estimating uncertainties implicit in both the way in which the code expansion ratio was estimated and the concern that some required algorithms might not have been included when data on the PLATO system was obtained.

A.5 Timing Estimates

Timing comparisons were made between the CYBER 73 and the PDP11/45 in order to insure that the PDP11/45 would not be CPU-bound when handling 24-32 terminals. An attempt was not made to accurately estimate the relative speeds of the two machines based upon the mix of instructions actually being executed on the PLATO system. Detailed information on the PLATO instruction mix was recorded by a system monitoring tool, but this data was not used since SofTech estimated that the precision required of the timing estimate would not justify the labor involved in making a more accurate estimate.

An iterative TUTOR calculation was the basis of the timing estimate since the CALC command accounts for about one-third of the execution time on the PLATO system and the CYBER 73 is likely to be particularly superior to the PDP11/45 in the area of numeric computation. The TUTOR code selected was:

```
DO      eval, index ← 0, 360, 2

UNIT    eval
CALC    c ← a x COS(2 x index)/b
```

where a, b, c and index are TUTOR student variables. The CALC command was assumed to be compiled to optimized machine code on each machine.

The results for each machine are:

	Code Length Words	Loop Initialization (μsec)	Time/ Iteration (μsec)	Total for 180 Iterations (μsec)
CYBER 73	11	8.9	38.6	6957
PDP 11/45 (with floating point processor)	30	14.0	152.0	27374

The calculated speed ratio is therefore 3.9, which indicates that the CPU power of a PDP-11/45 with a floating point processor is more than adequate for handling 24-32 terminals since a CYBER 73 could handle well over 150 terminals running the PLATO system if no authoring was allowed on the system.

Appendix C

DIGITAL EQUIPMENT CORPORATION QUOTATION

This appendix contains the hardware quotation submitted by Digital Equipment Corporation which fully identifies the hardware components. SofTech's review of the quotation with Digital Equipment indicated that the following changes should be made in the quotation:

Replacements	Effect on Quotation
MF11-U memory units should be replaced with MF11-UP parity memory units @ $6,300	$ 4,200
MM11-U expander core memory should be replaced with MM11-UP parity memory @ $5,600	$ 2,700
RJP04AA Disk and Controller with 44 million word capacity should replace RP11-CE moving head disk and RP03-AS disk pack drive	- $19,800
Add	
DA11-BD UNIBUS Link	$ 3,500
MF11-U 16K core memory to increase 11/40 memory to 64K	$ 4,900
Total Decrease	$ 4,580

digital EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS 01754
PHONE: AC 617 897-5111 TWX: 710-347-0212 - CABLE: DIGITAL MAYN, TELEX: 94-84-57

PLEASE REFER TO THIS QUOTATION NO. IN ALL CORRESPONDENCE AND ORDERS

DATE February 3, 1975
REFERENCE

NEAREST DIGITAL SALES OFFICE
Digital Equipment Corporation
235 Wyman Street
Waltham, Massachusetts 02154

TO Softech
460 Totten Pond Road
Waltham, Massachusetts 02154

Attn: Dr. John W. Brackett

Attn: Mr. Barry Nager

GENTLEMEN: THANK YOU FOR YOUR INQUIRY. WE ARE PLEASED TO QUOTE AS FOLLOWS

ITEM	QUANTITY	DESCRIPTION	UNIT PRICE	AMOUNT
		SYSTEM A		
1.	1	PDP-11/45-CW Central Processor with 32K parity core memory and hardware memory management. Includes power supply, line frequency clock, multi-device auto loader, power/fail restart, serial 30 CPS DECwriter terminal and control, and five training credits.	$ 37,570.00	$ 37,570.00
2.	3	MF11-U 16K word core memory and control with expansion capability to 32K.	$ 4,900.00	$ 14,700.00
3.	3	MM11-U 16K word expander core memory. Mounts on MF11-U.	$ 4,500.00	$ 13,500.00
4.	1	FP11-B Floating Point Processor.	$ 5,290.00	$ 5,290.00
5.	1	RJS04-BA First drive and controller, includes rack mounted RS04 fixed head disk drive with storage capacity of 512K of 16 bit words.	$ 18,000.00	$ 18,000.00
6.	2	RS04-AA 512 word fixed head drive.	$ 13,000.00	$ 26,000.00
7.	1	RP11-CE Moving Head Disk Unit.	$ 31,880.00	$ 31,880.00

TERMS: NET 30 DATE F.O.B. DIGITAL PLANT
FREIGHT CHARGES COLLECT
DELIVERY SCHEDULE
4 MONTHS A.R.O.

REMAINS FIRM FOR 60 DAYS FROM THE DATE OF THIS QUOTATION PRICE ... IN EFFECT ON THE DATE HEREOF ... MUST BE SIGNED IN MAYNARD, MASSACHUSETTS BY A DULY AUTHORIZED ... OF DIGITAL EQUIPMENT CORPORATION.

DIGITAL EQUIPMENT CORPORATION
By Barry Nager
Barry Nager - OEM Sales Rep.

C-2

CUSTOMER

SOFTECH

DEC 10 (101) 1079 -1072

TERMS AND CONDITIONS

The following are the Terms and Conditions under which Digital Equipment Corporation, hereinafter called DEC, sells its products. Where DEC sells products not appearing on its standard published price list the following Terms and Conditions are modified and supplemented by DEC's Computer Special Systems Terms and Conditions, incorporated herein by reference.

1. **TAXES.** Prices are exclusive of all sales, use and like taxes. Any tax DEC may be required to collect or pay upon the sale or delivery of the products shall be paid by the purchaser.

2. **DELIVERY.**

A. Delivery will be made F.O.B. DEC's plant with shipping charges to be paid by purchaser to carrier. Risk of loss shall pass to purchaser upon delivery by DEC to carrier. Purchaser hereby grants to DEC a security interest in the products as security for the performance by purchaser of all its obligations hereunder together with the right, without liability, to repossess the products with or without notice in the event of default of any such obligation.

B. DEC shall not be liable for any damages or penalty for delay in delivery due to failure to give notice of delay when such delay is due to the elements, acts of God, delays in transportation, delay in delivery by DEC's vendors or any other causes beyond the reasonable control of DEC. The delivery schedule shall be extended by a period of time equal to the time lost because of any such delay.

3. **SHIPMENT.** In the absence of specific instructions DEC will select the carrier but shall not thereby assume any liability in connection with shipment, and all such carriers shall be construed to be the agent of DEC.

4. **PAYMENT.** Terms are cash upon delivery, or at DEC's option net thirty (30) days from the date of delivery. If deliveries are authorized in installments, each shipment shall be paid for when due without regard to other scheduled deliveries. Notwithstanding the foregoing, shipment of the PDP-14 Repetitive Controller may be made by DEC without the Read Only Memory and may be invoked separately.

5. **PATENTS.** If notified promptly in writing of any action (and all prior claims relating to such action) brought against the purchaser, based on a claim that the equipment infringes a United States patent, DEC will defend such action at its expense and will pay the costs and damages awarded in any such action, provided that DEC shall have the sole control of the defense of any such action and all negotiations for its settlement or compromise. In the event that a final injunction shall be obtained against the purchaser's use of the equipment or any of its parts by reason of infringement of a United States patent, or if in DEC's opinion the equipment is likely to become the subject of a claim of infringement of a United States patent, DEC will, at its option and at its expense, either procure for the purchaser the right to continue using the equipment, replace or modify the same so that they become non-infringing, or grant the purchaser a credit for such equipment as depreciated and accepted by it upon its return. The depreciation shall be on equal amount per year over the lifetime of the equipment as established by DEC. DEC shall not have any liability to the purchaser under any provision of this clause if any patent infringement, or claim thereof, is based upon: (i) the use of the equipment in combination with other equipment and devices which are not made by DEC or (ii) the use of the equipment in practicing any process, or (iii) the furnishing to the purchaser of any information, data, service or applications assistance. The purchaser shall hold DEC harmless against any expense, judgment or loss for infringement of any patents or trademarks which results from DEC's compliance with the purchaser's designs, specifications or instructions. No costs or expenses shall be incurred for the account of DEC without the written consent of DEC. In no event shall DEC's total liability to the purchaser under or as a result of compliance with the provisions of this paragraph exceed the aggregate sum paid to DEC by the purchaser for the allegedly infringing equipment. The foregoing states the entire liability of DEC with respect to infringement of patents by the equipment or any part thereof or by their operation.

6. **WARRANTY AND REPAIR CHARGE POLICY.**

A. Computers, Computer Options and Controllers

1. All computers, computer options and controllers (except for the DECsystem-10, and as specified in subparagraph A.2. below) are warranted as indicated below for a period of three (3) months from date of installation. In the event that DEC is prevented by causes beyond its control, from properly installing the equipment, the period for this warranty shall be deemed to commence on the 30th day after delivery, or upon installation, whichever is sooner.

2. The PDP-14 Programmable Controller is warranted for a period of four (4) months from date of shipment.

B. Modules

1. All P, R, W, M, K and A modules shown in the then current applicable module catalog are warranted for a period of ten (10) years from date of shipment, providing parts are available. Handling charges of $5.00 or 10% of list price per unit, whichever is higher, will be applicable from one (1) year after delivery.

2. All Systems Modules, Laboratory Modules, High Current Pulse Equipment, G, S, H, Non Catalog Flip Chip Modules and Accessories are warranted for a period of one (1) year from date of shipment.

3. All modules must be returned, prepaid to DEC. Transportation charges covering the return of the repaired modules shall be paid by DEC within the contiguous forty-eight (48) United States and Canada and will be made on UPS or parcel post insured basis. Outside the contiguous forty eight (48) United States and Canada, purchaser shall pay all costs of shipping, customs clearance or any other related charges.

Please ship all modules to:

Digital Equipment Corporation
Logic Products Services - Repair Division
146 Main Street
Maynard, Massachusetts 01754

No modules will be accepted for credit or exchange without prior approval at DEC, plus proper Return Authorization Number (DEC RA #).

C. DEC's sole responsibility under the above warranties shall be at its option to either repair or replace any component which fails during the period of the applicable warranty due to a defect in workmanship and material, and provided the purchaser has promptly reported same to DEC in writing and DEC has, upon inspection, found such components to be defective. The purchaser must obtain shipping instructions for the return of any item under this warranty provision.

D. All above warranties are contingent upon proper use of the equipment. These warranties will not apply to: (i) adjustment, repair or parts replacement required because of accident, unusual physical or electrical stress, negligence, misuse, failure of electric power, air conditioning, humidity control, transportation or causes other than ordinary use; or (ii) the equipment has been installed by customer (except for the PDP-14 Programmable Controller or PDP-16) modified by the customer or where non-DEC approved components have been removed or altered, without prior written approval from DEC; or (iii) if the equipment has been dismantled and reinstalled by customer without the supervision of or prior written approval from DEC.

E. EXCEPT FOR THE EXPRESS WARRANTIES STATED ABOVE OR ON THE FACE HEREOF, DEC DISCLAIMS ALL WARRANTIES OF PRODUCTS, INCLUDING WITHOUT LIMITATION, ANY SOFTWARE SUPPLIED OR RECOMMENDED, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS and the stated express warranties are in lieu of all obligations or liabilities on the part of DEC for damages, including but not limited to special, indirect or consequential damages arising out of or in connection with the use or performance of the products.

7. **ACCEPTANCE OF COMPUTERS, COMPUTER OPTIONS AND CONTROLLERS.** Purchaser's acceptance shall occur upon successful completion of the acceptance tests as witnessed by the acceptance test report signed by a DEC representative. Alternance tests run by DEC, shall consist of DEC test procedures and programs applicable to the equipment. Acceptance tests shall be run at purchaser's site, if DEC is to install the equipment, otherwise at DEC's plant.

8. **INSTALLATION OF COMPUTERS, COMPUTER OPTIONS AND CONTROLLERS.** Unless otherwise agreed to in writing, the equipment shall be installed by DEC in any location within the contiguous forty eight (48) United States including the District of Columbia, Canada or a country in which DEC or a subsidiary of DEC has a service facility. The purchaser shall make available a suitable place of installation with all facilities in accordance with DEC's installation procedures. The purchaser shall furnish all labor required for unpacking and placing the equipment in the desired location. Notwithstanding the foregoing, DEC shall be under no obligation to install the equipment (i) unless the equipment and installation site is made available to DEC for installation within thirty (30) days from the date of delivery, and DEC has been so notified, or (ii) if the equipment has been modified without DEC's written approval and/or subjected to unusual physical or electrical stress, accident, neglect, misuse or other damage beyond the control of DEC. Notwithstanding the foregoing, installation of the PDP-14 Programmable Controller and PDP-16 shall be the sole responsibility of the purchaser.

9. **SOFTWARE LICENSE.** Software specified on the face hereof or specified in the then current price list as a component of a system is furnished to purchaser under a license for use on a single system and can be copied (with the inclusion of DEC's copyright notice) only for use in such system, except as may otherwise be provided in writing by DEC.

10. **PRODUCTION OF PDP-14 PROGRAMMABLE CONTROLLER READ-ONLY MEMORY.** As a prerequisite to the production of the Read Only Memory the purchaser shall submit to DEC two (2) verified Read Only Memory Source Tapes with a properly completed identifying label. Any changes made either by the purchaser to the Read Only Memory Source Tapes after submittal to DEC, or (ii) as a result of errors in the Read Only Memory Source Tapes, as determined by DEC, may result in additional charges to the purchaser.

11. **FIELD INSTALLATION OF OPTIONS.** Field installed options are subject to installation charges.

12. **SUBSTITUTIONS AND MODIFICATIONS OF SPECIFICATIONS.** DEC assumes the right to make substitutions and modifications in the specifications of products designed by DEC providing that such substitutions or modifications will not materially affect the performance of the products.

13. **MISCELLANEOUS.** A valid contract binding upon DEC will come only into being as of the time of delivery of written contract signed by an authorized agent of DEC at Maynard, Massachusetts, is dispatched to the purchaser by DEC. This contract is governed by the laws of the Commonwealth of Massachusetts. This contract is not assignable, any attempt to assign any rights, duties or obligations which may arise under this contract shall be void. All DEC rights and remedies, whether evidenced hereby or by any other contract, instrument or paper, shall be cumulative. It is expressly understood that in the event either party shall at any occasion fail to perform any term of this contract and the other party shall enforce that term or the failure to enforce on that occasion shall not prevent enforcement on any other occasion. Deviations from these terms and conditions are not valid unless confirmed in writing by an authorized officer of DEC at Maynard, Massachusetts. In no event will DEC be liable for special, indirect or consequential damages.

14. **MODIFICATION.** The foregoing terms and conditions shall prevail notwithstanding any variation from the terms and conditions of any order submitted by the purchaser.

SUPPLEMENTARY TERMS AND CONDITIONS OF SALE

Applicable to the Sale of Products for U.S. Government End-Use

The following clauses set forth in the Armed Services Procurement Regulations, as in effect on the date hereof, are incorporated herein when a Government contract number is specified on this contract. In the clauses marked with "*", the term "Contractor" shall mean Seller, and "Government" and "Contracting Officer" shall mean Buyer or the Government. If this contract is placed under a National Aeronautics and Space Administration prime contract, reference to ASPR clauses below or elsewhere in the order shall be deemed to have reference to the equivalent, if any, NASA PR clauses.

- 6-104.5 Buy American Act
- 7-103.2 Changes
- 7-103.13 Renegotiation
- 7-103.16 Contract Work Hours Standards Act
- 7-103.17 Walsh-Healy Public Contracts Act
- 7-103.18 Equal Opportunity Clause
- 7-103.19 Officials Not to Benefit
- 7-103.20 Covenant Against Contingent Fees
- 7-103.22 Authorization & Consent
- 7-103.23 Notice and Assistance
- 7-104.4 Notice to the Government of Labor Disputes*
- 7-104.15 Examination of Records

In addition to the above, DEC certifies that its facilities are non segregated.

C-3

REV 9/72

digital EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS 01754
PHONE: AC 617 897-5111 TWX: 710-347-0212 · CABLE: DIGITAL MAYN. TELEX: 94-84-57

PLEASE REFER TO THIS QUOTATION NO IN ALL CORRESPONDENCE AND ORDERS

DATE February 3, 1975

ITEM	QUANTITY	DESCRIPTION	UNIT PRICE	AMOUNT
8.	1	RP03-AS 20 million word moving head disk pack drive.	$ 20,000.00	$ 20,000.00
9.	1	TM11-EA 9 Channel Industry Compatible Tape Unit contains one TU10-E Tape Transport. Includes controller and cabinet.	$ 10,745.00	$ 10,745.00
10.	1	H960-DA Cabinet with single Extension Mounting Box.	$ 2,400.00	$ 2,400.00
11.	1	QJ580-AC RSX-11D real-time operating system software license for binaries delivered on DECtape.	$ 5,000.00	$ 5,000.00
		TOTAL		$ 185,085.00
		SYSTEM B		
1.	1	PDP-11/40-BA 11/40 central processor with 16K word core memory, ASR33 Teletype terminal and control (DL11-A) cabinet, and 4 training credits.	$ 15,500.00	$ 15,500.00
2.	1	KT11-D Memory management option.	$ 2,480.00	$ 2,480.00
3.	1	MF11-U 16K word core memory and control with expansion capability to 32K.	$ 4,900.00	$ 4,900.00
4.	1	MM11-U 16K word expander core memory. Mounts on MF11-U.	$ 4,500.00	$ 4,500.00
		TOTAL		$ 27,380.00

CUSTOMER

TERMS AND CONDITIONS

The following are the terms and conditions under which Digital Equipment Corporation, hereinafter called DEC, sells its products. Where DEC sells product not appearing on its standard published price list, the following Terms and Conditions are modified and supplemented by DEC's Computer Special Systems terms and conditions, incorporated herein by reference.

1 **TAXES.** Prices are exclusive of all sales, use and like taxes. Any tax DEC may be required to collect or pay upon the sale or delivery of the products shall be paid by the purchaser.

2 **DELIVERY.**

A. Delivery will be made F.O.B. DEC's plant with shipping charges to be paid by purchaser to carrier. Risk of loss shall pass to purchaser upon delivery by DEC to carrier. Purchaser hereby gives to DEC a security interest in the products as security for the performance by purchaser of all its obligations hereunder together with the right, without liability, to repossess the products with or without notice in the event of default of any such obligation.

B. DEC shall not be liable for any damages or penalty for delay in delivery or for failure to give notice of delay when such delay is due to the shipments, acts of God, delays in transportation, delay in delivery by DEC's vendors or any other causes beyond the reasonable control of DEC. The delivery schedule shall be extended by a period of time equal to the time lost because of any such delay.

3 **SHIPMENT.** In the absence of specific instructions DEC will select the carrier but shall not thereby assume any liability in connection with shipments, and shall not be construed to be the agent of the carrier.

4 **PAYMENT.** Terms are cash upon delivery or, at DEC's option net thirty (30) days from the date of delivery. If deliveries are authorized in installments, each shipment shall be paid for when due without regard to other scheduled deliveries. Notwithstanding the foregoing, shipment of the PDP-14 Repetitive Controller may be made by DEC without the Read Only Memory and may be invoiced separately.

5 **PATENTS.** If notified promptly in writing of any action (and all orders relating to such action) brought against the purchaser based on a claim that the equipment infringes a United States patent, DEC will defend such action at its expense and will pay the costs and damages awarded in any such action provided that DEC shall have the sole control of the defense of any such action and all negotiations for its settlement or compromise. In the event that a final injunction shall be obtained against the purchaser's use of the equipment or any of its parts by reason of infringement of a United States patent, or if in DEC's opinion the equipment is likely to become the subject of a claim of infringement of a United States patent, DEC will, at its option and at its expense, either procure for the purchaser the right to continue using the equipment, replace or modify the same so that they become non-infringing, or grant the purchaser a credit for such equipment as depreciated and accepted upon return. The depreciation shall be an equal amount per year over the lifetime of the equipment as established by DEC. DEC shall not have any liability to the purchaser under any provision of this clause if any patent infringement, or claim thereof, is based upon: (i) the use of the equipment in combination with other equipment and devices which are not made by DEC or (ii) the use of the equipment in practicing any process, or (iii) the furnishing to the purchaser of any information, data, service or applications assistance. The purchaser shall hold DEC harmless against any expense, judgment or loss for infringement of any patents or trademarks which results from DEC's compliance with the purchaser's designs, specifications or instructions. No costs or expenses shall be incurred for the account of DEC without the written consent of DEC. In no event shall DEC's total liability to the purchaser under, or as a result of compliance with, the provisions of this paragraph exceed the aggregate sum paid to DEC by the purchaser for the allegedly infringing equipment. The foregoing states the entire liability of DEC with respect to infringement of patents by the equipment or any part thereof or by their operation.

6 **WARRANTY AND REPAIR CHARGE POLICY.**

A. Computers, Computer Options, and Controllers

1. All computers, computer options and controllers (except for the DECsystem-10, and as specified in subparagraph A.2. below) are warranted as to materials, as set forth below for a period of three (3) months from date of installation. In the event that DEC is prevented by causes beyond its control, from properly installing the equipment, the period for this warranty shall be deemed to commence on the 30th day after delivery, or upon installation, whichever is sooner.
2. The PDP-14 Programmable Controller is warranted for a period of four (4) months from date of shipment.

B. Modules

1. All B, R, W, M, K and A modules shown in the then current applicable module catalog are warranted for a period of ten (10) years from date of shipment, providing parts are available. Handling charges of $5.00 or 10% of list price, per unit, whichever is higher, will be applicable from one (1) year after delivery.
2. All Systems Modules, Laboratory Modules, High Current Pulse Equipment, G, S, H, Non Catalog Flip Chip Modules and Accessories are warranted for a period of one (1) year from date of shipment.
3. All modules must be returned, prepaid to DEC. Transportation charges covering the return of the repaired modules shall be paid by DEC, within the contiguous forty-eight (48) United States and Canada and will be made on UPS or parcel post insured basis. Outside the contiguous forty eight (48) United States and Canada, purchaser shall pay all costs of shipping, customs clearance or any other related charges.

Please ship all modules to:

Digital Equipment Corporation
Logic Products Services - Repair Division
146 Main Street
Maynard, Massachusetts 01754

No modules will be accepted for credit or exchange without prior approval of DEC, plus proper Return Authorization Number (DEC RA #).

C. DEC's sole responsibility under the above warranties shall be, at its option, to either repair or replace any component which fails during the period of the applicable warranty due to a defect in workmanship and material, and provided the purchaser has promptly reported same to DEC in writing and DEC has, upon inspection, found such components to be defective. The purchaser must obtain shipping instructions for the return of any item under this warranty provision.

D. All of the warranties are contingent upon proper use of the equipment. These warranties will not apply to: (i) adjustment, repair or parts replacement as required because of accident, unusual physical or electrical stress, negligence, misuse, failure of electric power, air conditioning, humidity control, transportation or causes other than ordinary use; (ii) use of the equipment has been installed by customer, except for the PDP-14 Programmable Controller or PDP-16, modified by the customer or where the DEC equipment serial numbers have been removed or altered without prior written approval from DEC; or (iii) if the equipment has been disconnected and reinstalled by customer without the supervision of or prior written approval from DEC.

E. EXCEPT FOR THE EXPRESS WARRANTIES STATED ABOVE, DEC ON THE FACE HEREOF, DOES DISCLAIM ALL WARRANTIES ON DEC'S PRODUCTS, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTY OF MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE, AND THE STATED EXPRESS WARRANTIES ARE IN LIEU OF ALL OBLIGATIONS OR LIABILITIES ON THE PART OF DEC FOR DAMAGES, INCLUDING BUT NOT LIMITED TO SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE PRODUCTS.

7 **ACCEPTANCE OF COMPUTERS, COMPUTER OPTIONS AND CONTROLLERS.**

Purchaser's acceptance shall occur upon successful completion of the acceptance tests as evidenced by the acceptance test report signed by a DEC representative. Attendance tests run by DEC shall consist of DEC test procedures and programs applicable to the equipment. Acceptance tests shall be run at purchaser's site, if DEC is to install the equipment otherwise at DEC's plant.

8. **INSTALLATION OF COMPUTERS, COMPUTER OPTIONS AND CONTROLLERS.**

Unless otherwise agreed to in writing, the equipment shall be installed by DEC in any location within the contiguous forty eight (48) United States including the District of Columbia, Canada or a country in which DEC or a subsidiary of DEC has a service facility. The purchaser shall make available at a suitable place of installation with all facilities in accordance with DEC's installation procedures. The purchaser shall furnish all labor required for unpacking and placing the equipment in the desired location. Notwithstanding the foregoing, DEC shall be under no obligation to install the equipment (i) unless the equipment and installation site is made available to DEC for installation within thirty (30) days from the date of delivery, and DEC has been so notified, or (ii) if the equipment has been modified without DEC's written approval and/or subjected to unusual physical or electrical stress, accident, neglect, misuse or other damage beyond the control of DEC. Notwithstanding the foregoing, installation of the PDP-14 Programmable Controller and PDP-16 shall be the sole responsibility of the purchaser.

9. **SOFTWARE LICENSE.** Software specified on the face hereof or specified in the then current price list as a component of a system is furnished to purchaser under a license for use on a single system and can be copied (with the inclusion of DEC's copyright notice) only for use in such system, except as may otherwise be provided in writing by DEC.

10. **PRODUCTION OF PDP-14 PROGRAMMABLE CONTROLLER READ ONLY MEMORY.** As a prerequisite to the production of the Read Only Memory the purchaser shall submit to DEC two (2) verified Read Only Memory Source Tapes with a properly completed identifying label. Any changes made either by the purchaser to the Read Only Memory Source Tapes after submittal to DEC, or as a result of errors in the Read Only Memory Source Tapes, as determined by DEC, may result in additional charges to the purchaser.

11. **FIELD INSTALLATION OF OPTIONS.** Field installed options are subject to installation charges.

12. **SUBSTITUTIONS AND MODIFICATIONS OF SPECIFICATIONS.** DEC assumes the right to make substitutions and modifications in the specifications of products designed by DEC providing that such substitutions or modifications will not materially affect the performance of the products.

13. **MISCELLANEOUS.** A valid contract binding upon DEC will come only into being as of the time and date of written contract signed by an authorized agent of DEC at Maynard, Massachusetts, is dispatched to the purchaser by DEC. This contract is governed by the laws of the Commonwealth of Massachusetts. This contract is not assignable, any attempt to assign any rights, duties or obligations which may arise under this contract shall be void. All DEC rights and warranties, whether evidenced hereby or by any other contract, instrument or paper, shall be cumulative. It is expressly understood that in the event either party shall at any occasion fail to perform any term of this contract and the other party shall not enforce that term or the failure to enforce on that occasion shall not prevent enforcement on any other occasion. Deviations from these terms and conditions are not valid unless confirmed in writing by an authorized officer of DEC at Maynard, Massachusetts. In no event will DEC be liable for special, indirect or consequential damages.

14. **MODIFICATION.** The foregoing terms and conditions shall prevail notwithstanding any variation from the terms and conditions of any order submitted by the purchaser.

SUPPLEMENTARY TERMS AND CONDITIONS OF SALE
Applicable to the Sale of Products for U. S. Government End-Use

The following clauses set forth in the Armed Services Procurement Regulations, as in effect on the date hereof, are incorporated herein when a Government contract or lease is specified on this contract. In the clauses marked with (*), the term "Contractor" shall mean Seller, and "Government" and "Contracting Officer" shall mean Buyer or the Government. If this contract is placed under a National Aeronautics and Space Administration prime contract, reference to ASPR clauses below or elsewhere in the order shall be deemed to have reference to this equivalent, if any NASA PR clauses.

7-104.5 Buy American Act
7-103.2 Changes
7-103.13 Renegotiation
7-103.16 Contract Work Hours Standards Act
7-103.17 Walsh-Healy Public Contracts Act
7-103.18 Equal Opportunity Clause
7-103.19 Officials Not to Benefit
7-103.20 Covenant Against Contingent Fees
7-103.22 Authorization & Consent
7-103.23 Notice and Assistance
7-104.4 Notice to the Government of Labor Disputes
7-104.15 Examination of Records

In addition to the above, DEC certifies that its facilities are non segregated.

C-5

REV 9/72